Karl A. Stolleis

Candidate

Computer Science

Department

This thesis is approved, and it is acceptable in quality and form for publication: *Approved by the Dissertation Committee:*

Dr. Melanie M. Moses

Dr. Lydia Tapia

Dr. Rafael Fierro

The Ant and the Trap: Evolution of Ant-Inspired Obstacle Avoidance in a Multi-Agent Robotic System

by

Karl A. Stolleis

B.S., Texas A&M University, 1992

THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Computer Science

The University of New Mexico

Albuquerque, New Mexico

May, 2015

Dedication

To Elise, without whom my sanity would further be in question. And to all the friends I have made along the journey.

"Shall I refuse my dinner because I do not fully understand the process of digestion?" – Oliver Heaviside

Acknowledgments

I would like to thank my advisor, Dr. Melanie Moses, Dr. Lydia Tapia, fellow graduate students Joshua Hecker and Nick Malone without whose help this thesis would not have been possible. A thank you is also in order to all the members, past and present, of the Biological Computation Lab and the iAnt project. And finally thank you Cheryle Mako, Kurt Leucht and all the good folks at Kennedy Space Center.

The Ant and the Trap: Evolution of Ant-Inspired Obstacle Avoidance in a Multi-Agent Robotic System

by

Karl A. Stolleis

B.S., Texas A&M University, 1992

M.S., Computer Science, University of New Mexico, 2015

Abstract

Interest in swarm robotics, particularly those modeled on biological systems, has been increasing with each passing year. We created the iAnt robot as a platform to test how well an ant-inspired robotic swarm could collect resources in an unmapped environment.

Although swarm robotics is still a loosely defined field, one of the included hallmarks is multiple robots cooperating to complete a given task. The use of multiple robots means increased cost for research, scaling often linearly with the number of robots. We set out to create a system with the previously described capabilities while lowering the entry cost by building simple, cheap robots able to operate outside of a dedicated lab environment.

Obstacle avoidance has long been a necessary component of robot systems. Avoiding collisions is also a difficult problem and has been studied for many years. As part of moving the iAnt further towards the real-world we needed a method of obstacle avoidance. Our hypothesis is that use of biological methods including evolution, stochastic movements and stygmergic trails into the iAnt Central Place Foraging Algorithm (CPFA) could

result in robot behaviors suited to navigating obstacle-filled environments. The result is a modification of the CPFA to include pheromone trails, CPFA-Trails or CPFAT.

This thesis first demonstrates the low-cost, simple and robust design of the physical iAnt robot. Secondly we will demonstrate the adaptability of the the system to evolve and succeed in an obstacle-laden environment.

Contents

List of Figures

List of Tables

Glossary

iAnt	A small, low-cost swarm robotics research platform
CPFA	Central place foraging algorithm
CPFAT	Central place foraging algorithm modified to include stygmergic trail following
GA	Genetic algorithm, a heuristic search method approximating natural evolution

Chapter 1

Introduction

Interest in swarm robotics, especially systems modeled after insects, have garnered increasing attention in the last few years. The iAnt robot platform was created to test the ability of an ant-inspired robotic swarm to collect resources in an unmapped environment and return the resources to a central location [2]. This process, known as central-place foraging, is widely accepted as a viable behavioral strategy for robots working in mining, search-and-rescue or ordinance disposal [3, 4].

Other robotic systems were examined for cost and viability with regard to our desired research strategy [5]. Although several swarm systems are available commercially all proved too expensive and we felt construction of a new robot was warranted. The iAnt is composed of both commercial and custom designed parts with ease of assembly and low cost in mind. The total cost of a current iAnt is under $600 US dollars.

Also absent from the iAnt research, until now, was inclusion of obstacles in the environment and a study of how well the iAnt behavior could adapt in environments with obstacles. Obstacle avoidance, in robotics, has been researched for many decades and it is especially important for autonomous systems [6]. The bulk of this thesis is devoted to the work done incorporating obstacles and results from testing the modified iAnt behavior as

they navigate the new, more complex environment.

1.1 Contributions and Organization

Chapter 2 will provide the background for the iAnt robot platform and the physical design of the robot. The genesis of the robot was the need to find a low-cost, durable and simple robot for study of swarm robotics. Several systems were considered but cost was deemed too high. The iAnt is made up of custom-designed, laser cut acrylic parts as well as commercial off-the-shelf (COTS) parts where possible. The detailed drawings and parts lists are presented in the Appendices and complete instructions are available online.

Chapter 3 is devoted to work done to incorporate obstacles and obstacle avoidance into the iAnt central-place foraging algorithm (CPFA). Avoidance of obstacles, for robots, is important and a great deal of research has been devoted to the topic. First I will demonstrate the ability of the CPFA to evolve behaviors via genetic algorithm (GA) that can maximize fitness while operating in obstacle strewn environments. Not only is fitness maximized but collision calls, made by the robots, is decreased. Secondly I will show that the addition of stygmergic trails to the CPFA, resulting in CPFA-Trails (CPFAT), is able to further increase fitness, reduce obstacle avoidance calls and succeed against a pathological case which defeated the CPFA. The work in this chapter was co-authored by Dr. Melanie Moses and Joshua Hecker.

Chapter 4 is the conclusion and future work.

Chapter 2

The iAnt Robot Platform

2.1 Introduction

When we began work on the iAnt our initial intent was to purchase robots or a system which met our needs, not to build our own. The E-Puck and the Kilobot were both considered as both met the definition of swarm robots but cost was prohibitive [7, 8]. Our attention then shifted to the idea of constructing our own robot while maintaining the following capabilities:

- Lower the initial cost, per robot of setting up a multi-agent system
- Utilize as many commercial off-the-shelf (COTS) parts as possible
- Make the design simple to construct for other researchers or educators
- Provide all design and construction details as “open-source”

2.2 The First Generation iAnt Robot

The first generation of the iAnt began in the summer of 2011 and although many changes have been made since, it is relevant to show the basis of the iAnt and the on-board sensors. The initial version used an Arduino micro-controller for all computing tasks, a Global Positioning System (GPS) module for localization, compass for heading, 802.11b wireless module for communication, ultrasonic range-finder and radio-frequency identification (RFID) reader for resource collection. The selected chassis was a commercial off-the-shelf product chosen for its size and cost. A complete document detailing the construction and design considerations is presented in Appendix A.

2.3 The Current Generation iAnt Robot

The current (2014) version of the iAnt is an evolution in and of itself, from the first version. Many changes took place including the use of an Apple iPod for increased computing power and inclusion of cameras, removal of the GPS for indoor use, removal of the RFID and substitution of quick-response (QR) codes to simulate resources and a completely new chassis designed from laser cut acrylic. The Arduino micro-controller was retained, along with the compass, range-finder and motor driver ”shield.”

Work began with construction of a three-dimensional computer-aided design (3D-CAD) model of the necessary parts to test dimension and fitment, Fig.2.1.

To allow use of both cameras, on board the iPod, a set of pieces were designed to mount the iPod horizontally and using an angled mirror mounted above the upward-facing camera. The result is one camera able to look down for resource detection and the other to look ahead for navigation, Fig.2.2,2.3,2.4.

The original chassis was scrapped in favor of a wheeled-chassis incorporating similar

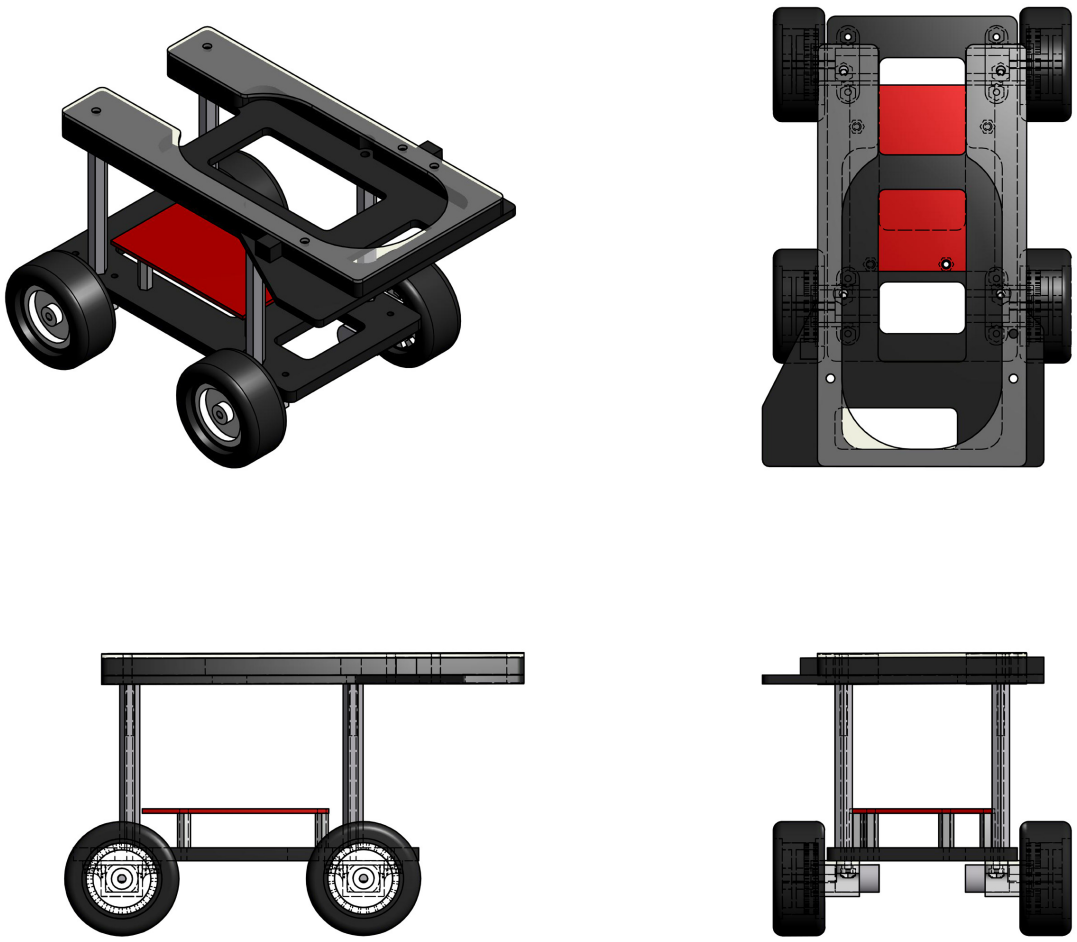


Figure 2.1: 3D CAD model of 4th generation iAnt platform assembly.

motors but having lower gear ratios. The tracked chassis showed poor motor life with a failure happening approximately 10-12 hours of run time. The wheeled configuration has seen motor failures decrease to as little as every 60-70 hours of run time. In addition to use of lower gear-ratio motors a small cap was added to seal the motor gears from damage due to sand to grit and likely contributed to the extended motor life. The wheeled chassis also allows for faster motor replacement in the event of a failure.

Additional battery capacity, over the first generation, is due to use of a 7.4 volt, 1100

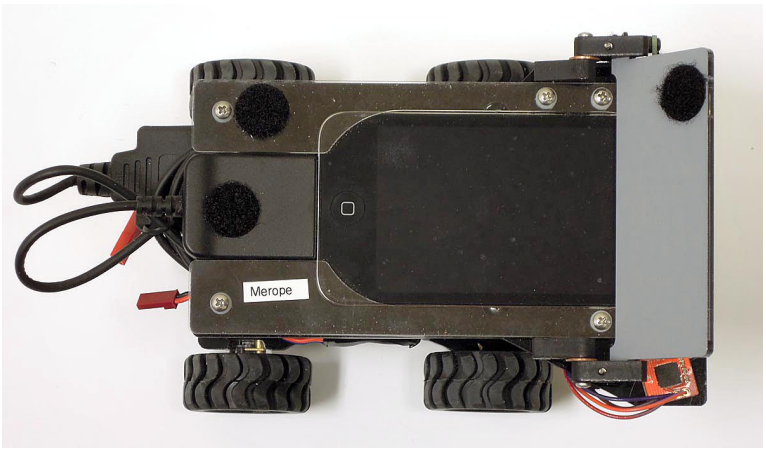


Figure 2.2: Top view of assembled 4th generation iAnt.

milli-amp/hour (mAh) lithium polymer battery.

The final result is a robot that maintained a total cost of approximately $550.00 and utilizes easily replaceable parts made from flat, cast acrylic material. The final assembly is also one easily modified to allow for future changes to either design or capability.

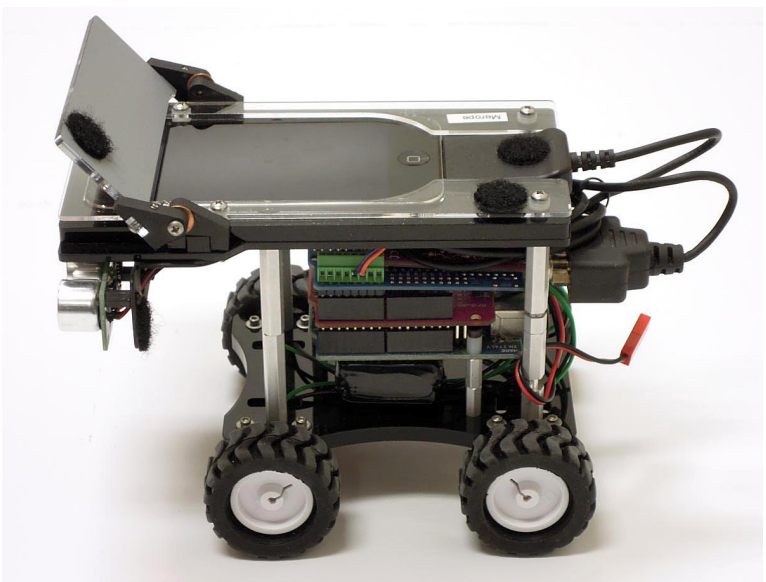


Figure 2.3: Side view of assembled 4th generation iAnt.

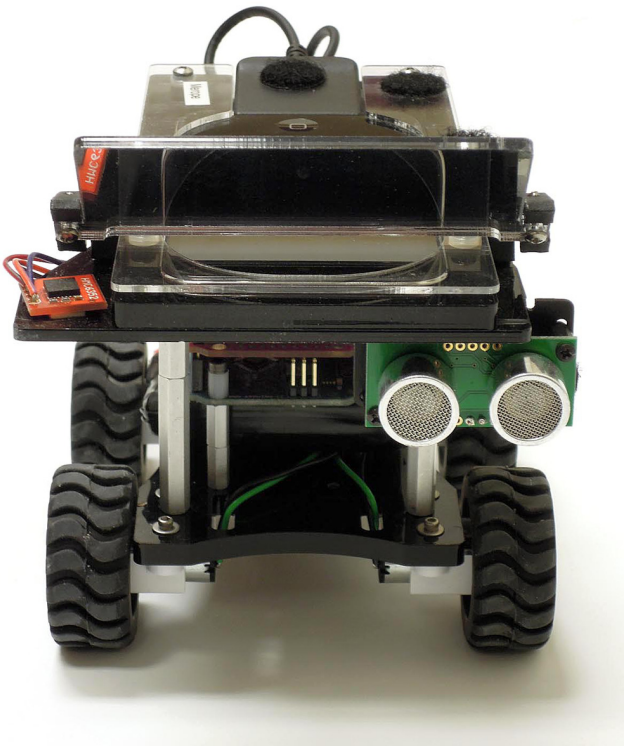


Figure 2.4: Front view of assembled 4th generation iAnt.

The dimensioned drawings for the acrylic parts are included in Appendix B. The full parts list, along with suppliers and costs are included in Appendix C.

Chapter 3

Evolution of Ant-Inspired Obstacle Avoidance in Swarming Robots

3.1 Abstract

In this chapter we demonstrate that a simple robotic swarm can use a genetic algorithm (GA) to achieve autonomous navigation in highly cluttered environments. We also show the GA can evolve effective strategies for coping with random obstacle placements using a central place foraging algorithm (CPFA). The CPFA combines stochastic movements, insect-inspired obstacle avoidance and simple communication among robots. We modify the CPFA by allowing simulated robots to communicate with stygmergic trails in CPFA-Trails (CPFAT). Both the CPFA and CPFAT can evolve foraging strategies to collect resources in the presence of a high density of randomly placed obstacles. CPFAT additionally succeeds against a classic bug trap, where the CPFA fails. These methods are simple to implement, run in real-time and need no global knowledge of the environment. This chapter was authored by myself and in collaboration with by Melanie E. Moses and Joshua Hecker (Department of Computer Science, University of New Mexico) and submitted for

publication.

3.2 Introduction

If swarming robots are to leave the controlled environment of research laboratories, where many currently navigate, they will have to deal with real-world complexities, not least of which is avoiding collisions [6, 9]. Obstacle avoidance has long been recognized as a challenging problem, and collisions with objects can cause damage to both the objects and the robots, and impair the robots' ability to complete their tasks [6].

The iAnt swarm has demonstrated flexible, error-tolerant foraging using the Central Place Foraging Algorithm (CPFA) optimized by a genetic algorithm (GA) [2]. The foraging task is also known to be difficult and one well suited to multi-agent systems [10]. Real-world problems, similar to foraging, include terrestrial mining and search-and-rescue [3, 11]. The iAnt robot swarm forages using stochastic movements and a simple communication strategy, but until now, has not attempted to navigate environments containing obstacles.

The iAnt swarm utilizes a GA to evolve autonomous behaviors that are automatically tuned to different environments [1]. In this work we show the GA can evolve behaviors to forage successfully in the presence of obstacles. We additionally modify the CPFA to include stygmergic trails and a simple, insect-inspired obstacle avoidance algorithm. We call the resulting, novel method CPFA-Trails (CPFAT). We test the CPFA and CPFAT against environments with randomly placed obstacles and a pathological trap used widely to test performance of path planning algorithms [12–14]. The GA is able to evolve effective strategies for coping with randomized obstacles in both the CPFA and CPFAT by producing unique parameter sets appropriate to each method. Finally we demonstrate that the CPFAT method is able to succeed against the insect trap arrangement where the CPFA

method is not. The CPFAT retains all the hallmarks of the original CPFA in its ability to achieve flexible, real-time foraging in the absence of global knowledge [1].

3.3 Background

3.3.1 Obstacles and Collision Avoidance

Obstacle avoidance has been studied for over three decades in robotics [6]. The reasons for avoiding obstacles are simple. As a robot moves from the lab to more uncertain environments encounters with obstacles and possible damage to both robot and obstacle become more probable. The issue was addressed in early designs of possible planetary rovers by NASA [15–17]. In 1979 it was shown that generalized path-planning problems (i.e., the Mover's Problem) belong to the complexity class PSPACE-hard [18].

3.3.2 Path-Planning without Global Knowledge

Path planning algorithms can be divided into two distinct categories, those having global knowledge of the space and those that do not [19]. Often local planning is simply avoiding an obstacle without regard to overall navigation path [6, 19].

Bug algorithms are a family of path planning algorithms that typically utilize only local knowledge except for the location of the robot's final destination or sink [20, 21]. The body of research surrounding bug algorithms is deep, varied and many improvements have been proposed [22–27].

Taking the example of biological inspiration further, ant behavior has been used as model strategy for obstacle avoidance and path planning [28–32]. The main focus of this research lies in the stygmergic trails of pheromones laid by ants as a method of sharing

information about the environment [33–39].

A shortcoming of bug algorithms lies in the classic "bug trap," Fig. 3.1(L), often used as a pathological test case for planning algorithms [12–14]. The bug trap takes its name from actual traps used on insects which are easy to enter but difficult to escape [40]. Research has also shown that randomization, within the path planning task, can allow robots to escape bug traps [41].

3.3.3 The Foraging Task

Ant behavior has also been used as a model for behavior of multi-robot systems [4, 9, 42–45]. Ants foraging for food search a space, collect resources and return those resources, typically to a central location. This particular behavior would be well suited to robots collecting *in situ* resources in an extra-planetary setting, terrestrial mining, search-and-rescue and security tasks [3, 11, 46]. The collection of robots performing the foraging are often referred to as a swarm. Swarms are usually characterized by cooperation, local communication and lack of centralized control. Swarm robotic algorithms require error-tolerance and flexibility, particularly because the robots need to be small and relatively inexpensive, in order to make large swarms economically feasible [47, 48].

3.3.4 The iAnt and Evolution of Autonomous Behavior

Previously, we observed and modeled ants foraging in natural environments [49], parameterized those models, used a GA to maximize resource collection rates for different resource distributions [38, 50] and instantiated those foraging parameters in iAnt robot swarms [2, 51, 52]. We briefly describe the CPFA below by summarizing the longer description in [1]. Parameters are listed in Table 3.1.

In [1] the GA-tuned CPFA evolved appropriate solutions to various environmental con-

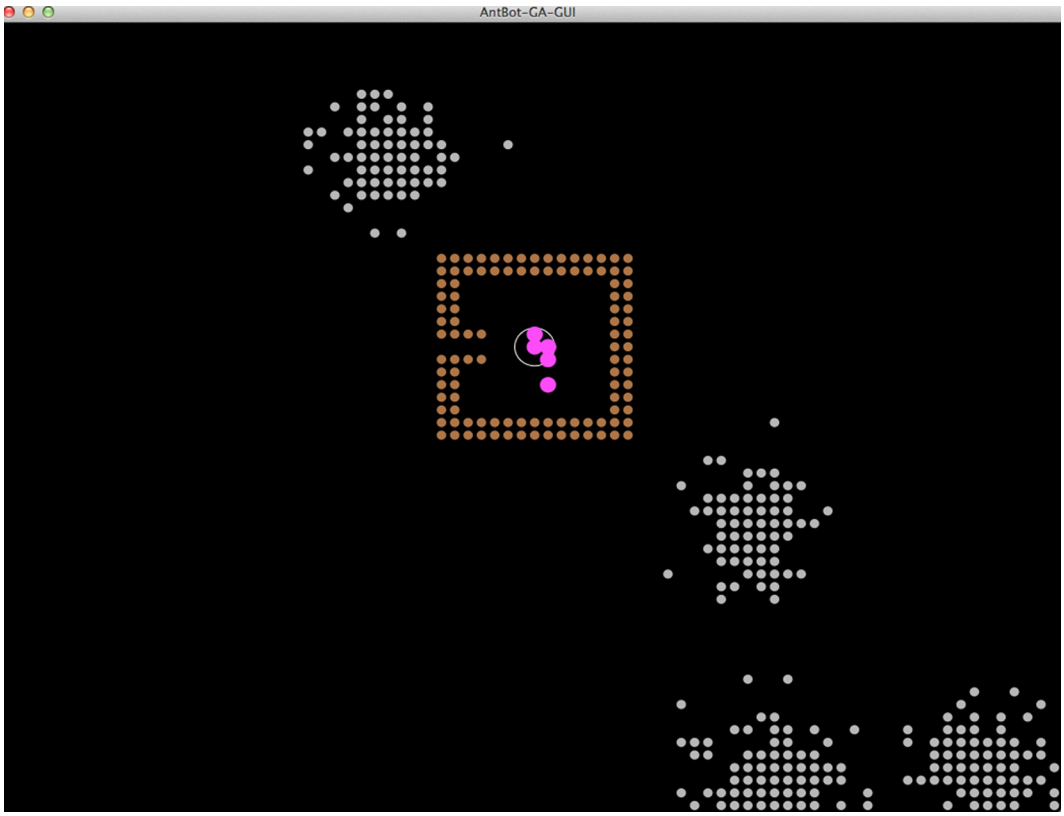


Figure 3.1: The iAnt simulation environment including the bug trap obstacle arrangement and a resource distribution of 256 targets divided into four discrete clusters. Robots are magenta (center) obstacles are brown and resources are gray. The nest is in the center of the trap.

Table 3.1: Evolved Parameters Controlling iAnt Behavior

Parameter	Initialization Function
Informed Search Decay Rate	Exponential(5)
Pheromone Decay Rate	Exponential(1)
Pheromone Laying Rate	Uniform(0, 20)
Search Give-Up Probability	Uniform(0, 1)
Pheromone Following Rate	Uniform(0, 20)
Travel Give-Up Probability	Uniform(0, 1)
Uninformed Search Variation	Uniform(0, 4π)

ditions, for example: *i*) increased communication when information was reliable and resources were highly clustered, *ii*) more individual memory when cluster sizes were variable, and *iii*) greater dispersal with increasing swarm size. The GA was able to evolve foraging strategies tolerant of real-world sensing and navigation error, flexible with respect to various resource distributions, and scalable to large numbers of agents [1].

Fig. 3.2 shows the states of the CPFA. The appropriate set of movement and communication behaviors depends on various features of the foraging problem, most notably here, how resources are distributed in the environment.

The CPFA is implemented as a state machine with probabilistic transitions. The parameters in Table 3.1 govern the transitions. Each robot begins foraging from a central nest by setting a new search location. Robots traveling to a random location with no prior information search using an uninformed correlated random walk where the degree of turning in the walk is governed by the parameter *Uninformed Search Variation*. Robots traveling to a previously found resource location use an informed random walk to thoroughly search the area. The informed walk has more variation in turning angles. That variation decays over time into an uninformed walk at a rate determined by *Informed Search Decay Rate*. When a robot locates a resource it collects the resource, records a count of neighboring resources and returns to the nest. The decision to lay and follow pheromones are made based on the parameters *Pheromone Laying Rate* and *Pheromone Following Rate*. Pheromones decay exponentially over time controlled by the parameter *Pheromone Decay Rate*.

The specific implementation of several CPFA behaviors are particularly relevant to the work presented here. First, in the original CPFA which was designed assuming obstacle-free environments, pheromones are simply transmitted as way-points rather than complete trails from the nest to the resource location. Second, during the search phase, robots move using either an informed or uninformed random walk. In the uninformed random walk, at each time step a robot chooses a new travel direction from a Gaussian probability distribution, where the mean is the current robot heading and the standard deviation is a

parameter evolved by the GA. After choosing the new heading the robot advances one cell. The informed random walk is similar, but with a different standard deviation around the robot's heading. The GA evolves a larger standard deviation for the informed walk, increasing the Brownian motion and in turn resulting in a more complete search of the region surrounding the robot's destination. Third, when the iAnt attempts to travel in a straight line, either returning to the nest or traveling to a previous resource site, there is a stochastic component. At each step, a robot is most likely to choose an optimal, shortest path but the robot can choose a non-optimal step at any point during the travel.

Although the CPFA can evolve effective solutions for different resource distributions, here we consider only the highly clustered distribution, Fig. 3.1. Evidence suggests that most natural resource distributions are clustered [53, 54]. Pheromone communication is particularly important for clustered resources, because pheromones recruit new robots to the discovered clusters [1].

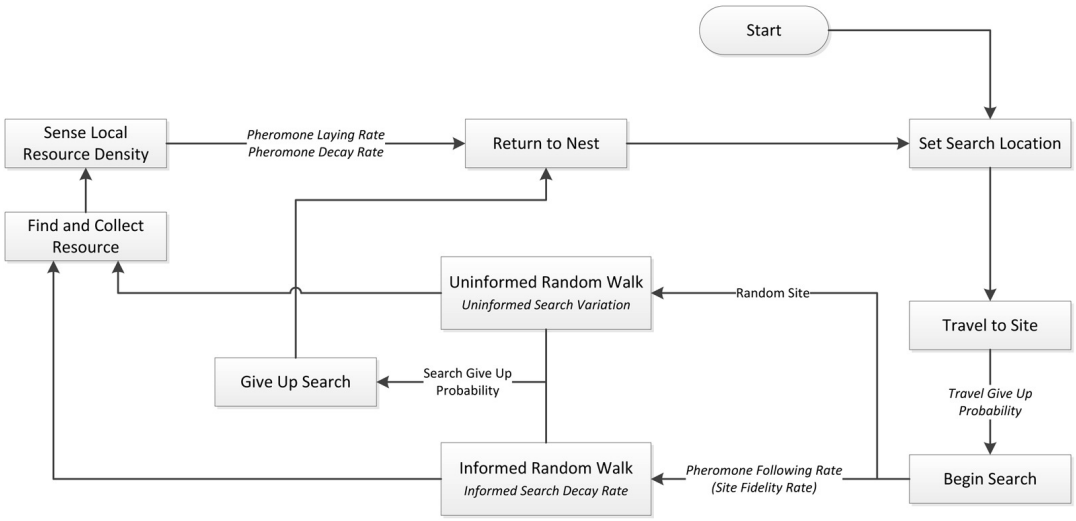


Figure 3.2: The iAnt CPFA state diagram. Parameters from Table 3.1 are in italics. Modified from [1].

3.4 Methodology

3.4.1 Central-Place Foraging and Evolutionary Algorithms

Following methods detailed in [1] we use a GA to evolve a population of CPFA parameters that maximizes the foraging efficiency of simulated robot swarms evaluated in an agent-based model. These parameters control the sensitivity threshold for triggering CPFA behaviors, likelihood of transitioning from one state to another, and length of time spent in a given state.

3.4.2 Distribution of Obstacles

We use four types of obstacle arrangements in experiments. The first arrangement utilizes a 2 *x* 2 cell square obstacles placed randomly, referred to as *Random*, in the arena. A one-cell wide corridor around the outer perimeter and a four-cell square around the central nest is left free of obstacles. Numbers of obstacles range from a baseline of zero to a maximum of 1000. It should be noted that high densities of obstacles rarely subdivided the space completely and did occasionally form "bug traps," from which robots struggled to escape. Fig. 3.3 illustrates the maximum obstacle density.

The second obstacle arrangement is a previously described bug trap, referred to as *Trap*, surrounding the central nest [12]. The inner dimension is 12 *x* 12 cells with a 1-cell wide opening on the left side of the trap. Fig. 3.1 shows the trap arrangement. Experiments with the trap environment did not include any additional obstacles.

The third arrangement, shown in Fig.3.4 (L), has large obstacles separating the nest from the resource clusters with three narrow passages, referred to as *Passages*, between the two regions (top, bottom and center). The passages are a single grid cell in width. The fourth arrangement, shown in Fig.3.4 (R), has walls, referred to as *Walls*, separating

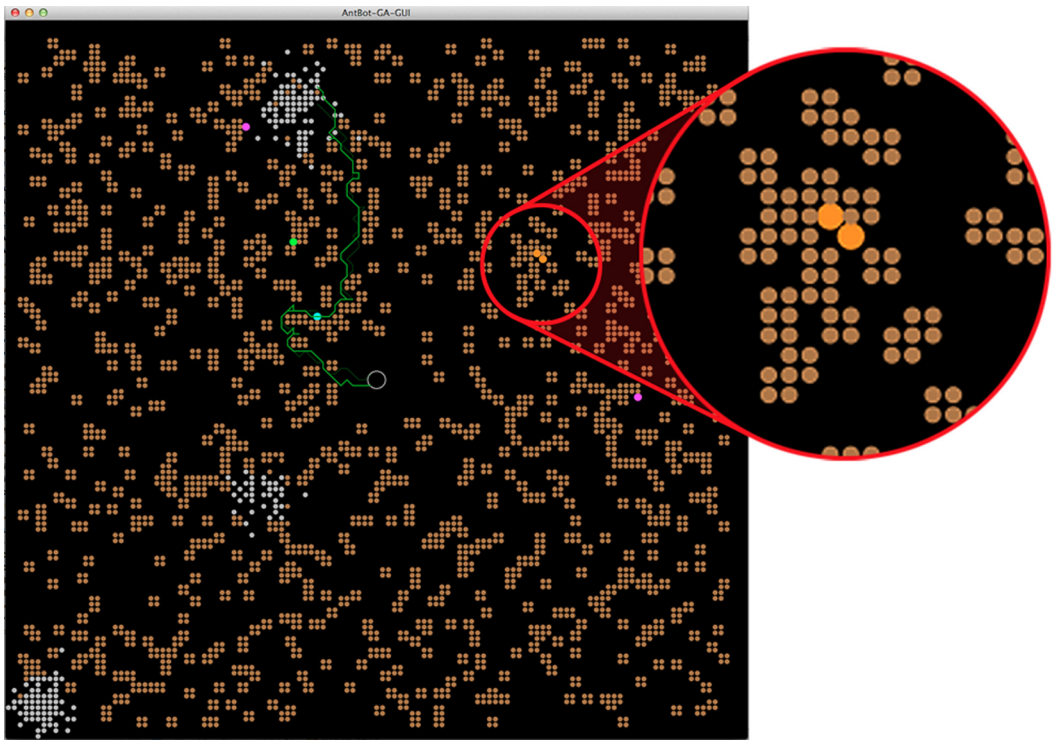


Figure 3.3: Robot search environment with 1000 randomly distributed obstacles and clustered resource distribution. Inset shows two robots ensnared in a bug trap and unable to escape. Pheromone trails to resource cluster are visible, top-center. Obstacles are brown, resources are gray and robots are green, orange, purple or cyan. The nest is white circle in center.

the environment into five equal regions. Each wall contains two, single-cell openings randomly placed in the walls.

3.4.3 Obstacle Avoidance Algorithm

Our obstacle avoidance algorithm belongs to the family of bug algorithms. The method is similar to one of the simplest, "Dist-Bug" [21, 55]. When a robot decides to move to a neighboring cell it first checks to see if the cell is obstructed. If obstructed, the robot checks neighboring cells in a clockwise direction until a clear cell is found. The robot then

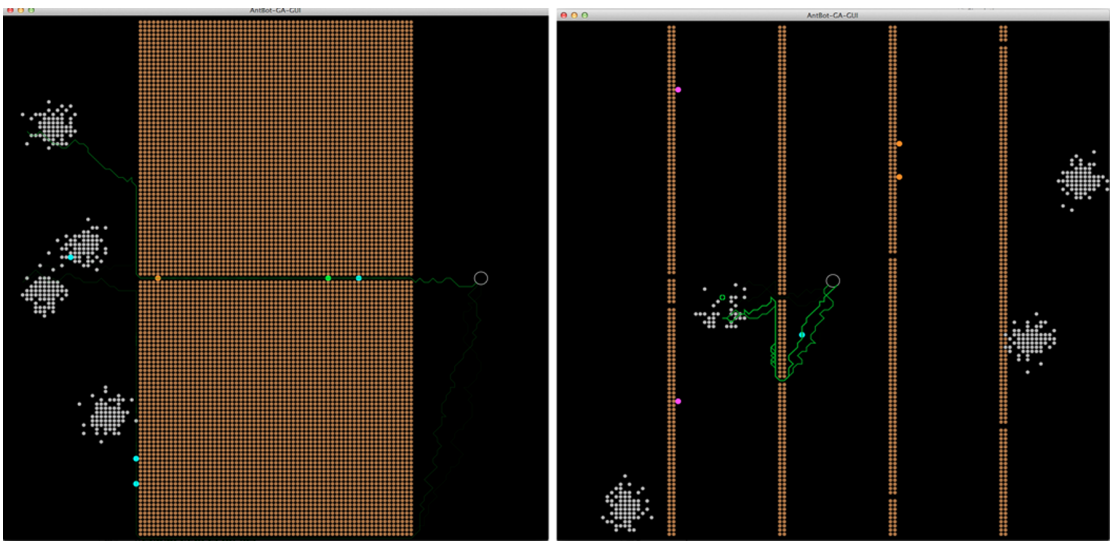


Figure 3.4: Additional obstacle arrangements used in testing. (Left) *Passage* arrangement. (Right) *Walls* arrangement. Obstacles are brown, resources are gray and robots are green, orange, purple or cyan. The nest is the white circle.

moves to the clear cell and returns to its previous task. If performing a random walk there is no guarantee the robot will not turn immediately back to the obstacle and have to repeat the procedure. Failure occurs when the robot has a destination lying beyond its eight-cell neighborhood. After moving to a clear cell the robot will turn back towards the destination and is guaranteed to run into the trap walls again. In the iAnt algorithm a distant waypoint is used to path-plan when the robot is utilizing information, i.e. traveling to nest or previous resource discovery site. The primary difference between the Dist-Bug algorithm and the one employed here is the randomness incorporated into the basic movement even when approximating straight line travel [50].

3.4.4 Pheromone Trails

In CPFAT we introduce a trail consisting of a list of actual grid points the robot moves through while returning to the nest with a resource. The robot uses the iAnt straight line

approximation algorithm to travel between its current position and the nest. When a robot departing the nest chooses to follow pheromones it is provided with one of the available trail lists. The robot will then follow the provided trail cell-by-cell duplicating the path of the robot that laid the path.

The provided paths are not expected to be optimal with regard to shortest distance but are guaranteed to provide a complete and collision free path to the trail terminus. Research indicates that ants may minimize travel time, rather than path length, to optimize trails so the traditional metric of optimality may not be valid [56].

Trails are only followed when leaving the nest and returning to a previously discovered resource. Trails are not used when returning to the nest nor are they used during random search. Trail use in the CPFAT is evolved by genetic algorithm in exactly the same way all CPFA parameters are evolved [1].

3.4.5 Experimental Set-up

Experiments were conducted in the iAnt simulation environment. The simulation allows for selection of resource distribution, number of robots, size of search environment, obstacle number and distribution and inclusion of error. Robots do not detect other robots as obstacles. Collisions only occur with static obstacles placed in the environment. The following conditions were used for for all experiments:

- 256 resource targets or tags divided into four randomly placed clusters
- Area equal to 100 m^2 discretized into 125 x 125 cells
- Mann-Whitney U test used for statistical significance

and the following conditions were used to evolve CPFA parameters with the GA:

- 100 swarms evaluated for each generation of evolution
- 50 generations of evolution
- GA employs elitism
- 10 independent iterations of the GA determines the optimal parameter set to evaluate
- 1000 evaluations with best parameters conducted post evolution

3.5 Results

3.5.1 Random Obstacles

CPFA Obstacle Evolution

Fig. 3.5(L) shows fitness of the CPFA with increasing obstacle density. The three lines plotted represent full evolution of parameters against each obstacle density (blue), parameters evolved with no obstacles evaluated against increasing obstacle density (red) and the parameters evolved against 1000 obstacles and tested with increasing density (green). Behaviors evolved for each specific density (blue) shows better performance, on average, than behaviors evolved from a single obstacle density. Swarms with obstacle-evolved parameters collected 4.0% more resources than the parameters assuming 1000 obstacles ($p = 0.04$). Swarms with obstacle-evolved parameters also show a marginally significant 3.1% increase in foraging performance with 1000 obstacles, over the parameters evolved assuming no obstacles ($p = 0.05$).

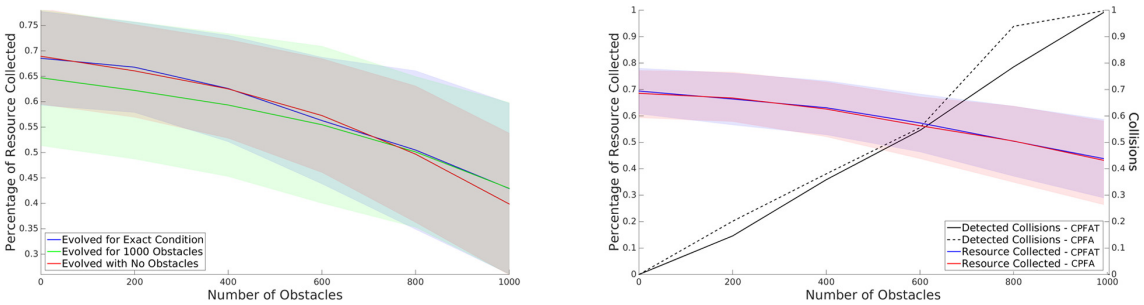


Figure 3.5: (Left) Results of testing three different evolved parameter sets against increasing obstacle density and using CPFA way-points. The blue line shows parameters evolved for the particular obstacle density on which they were tested. The green line shows parameters evolved for only maximum density but tested against increasing density. The red line shows parameters evolved for no obstacles tested against increasing density. (Right) Evolved fitness of both CPFA (red) and CPFAT (blue) are shown against increasing obstacle density. The black lines indicate the fraction of collision avoidance calls made by all robots for a given obstacle density. Shaded regions represent the 25% and 75% quantiles. The right axis shows collisions relative to the total number of collisions from the random obstacles.

CPFA & CPFAT Fitness

Fig. 3.4(R) shows overall fitness for CPFA (red) and CPFAT (blue) versus obstacle density. CPFAT shows a marginally significant increase of 2.0% compared to CPFA ($p = 0.05$).

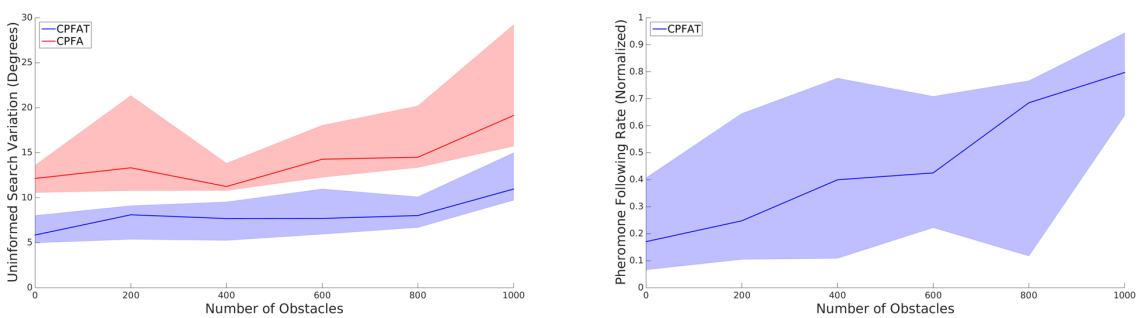


Figure 3.6: (Left) Uninformed search variation for both CPFA (red) and CPFAT (blue) show correlated increases with respect to increasing obstacle density. The values are presented as the range of degrees around the robot's current heading which may be chosen for the next walk step. (Right) Pheromone following rate for CPFAT shows a correlated increase with respect to obstacle density. The values are medians of the ten best, evolved parameter sets.

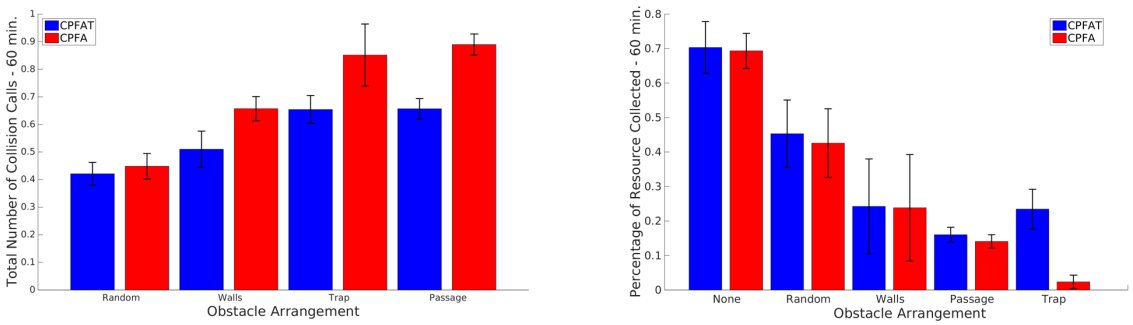


Figure 3.7: (Left) The number of total collision avoidance calls made by the CPFA (red) and CPFAT (blue) evaluated against the obstacle arrangements. (Right) The total fitness values for both CPFA (red) and CPFAT (blue) evaluated against three possible obstacle arrangements. Error bars represent the 25% and 75% quantiles. Refer to section IIIb for obstacle description. The random arrangement included 1000 individual obstacles.

CPFA & CPFAT Collision Calls

Fig. 3.5(R) shows the total number of collision calls made by both the CPFA (dashed) and CPFAT (black). Collision calls, with increasing obstacle density, showed a rate of increase faster than rate of fitness decrease. Thus, while the addition of trails with CPFAT made a slight difference in the number of resources collected, trails substantially decreased the number of collision calls.

Increasing Turn Angle

Fig. 3.6(L) shows the only parameter that evolved a significant change as obstacle density increased: the uninformed search correlation. Both the CPFA (red) ($p < 0.01$) and CPFAT (blue) ($p < 0.01$) show increases in possible turning angles used at each step of the uninformed random walk. The uninformed search correlation values for the CPFA are approximately 8.2 degrees greater than for CPFAT.

Table 3.2: Evolved Parameters For Each Obstacle Environment

Parameter	None CPFA	None CPFAT	Random CPFA	Random CPFAT	Walls CPFA	Walls CPFAT	Passage CPFA	Passage CPFAT	Trap CPFA	Trap CPFAT
Informed Search Decay Rate	0.1890	0.1732	0.1388	0.0795	0.2392	0.0858	0.4289	0.0382	0.0983	0.0626
Pheromone Decay Rate	0.0316	0.0144	0.0159	0.0345	0.0230	0.0250	0.0049	0.0127	0.1524	0.0119
Pheromone Laying Rate	1.4052	3.6324	4.5668	1.6652	0.7846	0.3421	1.4948	1.5240	4.2519	2.3580
Search Give-Up Probability	0.0061	0.0098	0.0047	0.0130	0.0097	0.0061	0.0060	0.0044	0.0029	0.0037
Pheromone Following Rate	1.6494	1.5053	1.5508	19.8645	3.6803	19.4112	5.3931	13.2560	0.7897	19.3037
Travel Give-Up Probability	0.1353	0.1253	0.3560	0.1442	0.2508	0.1333	0.0025	0.0031	0.8935	0.4139
Uninformed Search Variation	0.2640	0.1996	0.3678	0.4554	0.8766	0.8293	0.4059	0.5084	0.4683	0.5496

CPFA & CPFAT Unique Parameters

Fig. 3.6(R) shows the other parameter with a significantly positive increase as obstacle density increases using CPFAT: the pheromone following rate ($p < 0.01$). The final value indicates a high probability that pheromone trails will be followed when available. In contrast the CPFA did not show a significant increase in pheromone following (NS, $p = 0.08$) as obstacle density increased. Table 3.2 shows a list of best evolved parameters, for different environments. Note that with no obstacles, pheromone following rate is similar, between CPFA and CPFAT, but with all other cases the pheromone following rate is substantially different between the two methods. The high pheromone following rate for CPFAT indicates a strong preference to use stygmergic trails in the presence of obstacles.

3.5.2 The Trap, Passage and Walls

CPFAT Decreases Collision Calls

Fig. 3.7(L) shows total number of collision calls made by the CPFA (red) and CPFAT (blue) versus maximum obstacle density and the other three obstacle arrangements. As previously shown, the decrease with randomly placed obstacles is significant ($p < 0.01$). The difference in collision calls with the walls, narrow passage and the bug trap are significant ($p < 0.01$) and greater than with random obstacles. Each parameter set was evolved for the individual situations.

CPFAT Succeeds and CPFA Fails

Fig. 3.7(R) shows the effect on fitness for both CPFA (red) and CPFAT (blue) as obstacles increase. With no obstacles both methods were similar and showed no significant difference ($p = 0.66$). With 1000 random obstacles the difference in the number of collected resources became significant while still relatively small ($p < 0.01$). With the bug trap the CPFAT was still able to gather 22.3% of resources while the CPFA was only able to collect 2.0% ($p < 0.01$). The *Walls* and *Passage* arrangements showed significant differences between CPFA and CPFAT ($p < 0.01$) although the difference between the two methods was not as marked as with the trap. Each parameter set was evolved for the case in which they were evaluated.

GA Prefers Trails for CPFAT

Table 3.2 displays the best evolved parameter values for each obstacle arrangement, from the ten, independent evolutions. Note that in all cases the evolved parameter for pheromone following rate is much higher in all CPFAT cases. The GA strongly prefers use of trails when available while relying more on site fidelity or individual memory for the CPFA.

3.6 Discussion

Our initial expectation was the CPFAT would always outperform the original CPFA. However, CPFAT resulted in only minor foraging improvement for randomly placed obstacles, even with very high obstacle density. Fig. 3.5 shows the GA is able to tune parameters to each unique obstacle density and modestly improve foraging in the presence of obstacles. However, when comparing fitness and collision calls between the CPFA and the CPFAT there were significant gains made by the GA using trails. Not surprisingly, the naïve CPFA which evolves behaviors in environments with no obstacles, collects fewer resources as the

number of obstacles increases during evaluation. Swarms collect 42% of resources with 1000 obstacles compared to 68% of resources with no obstacles. Interestingly, the GA evolves a unique parameter set or different strategy, when robots lay trails rather than the way-points of the original CPFA. Refer to Table 3.2 for evolved values.

Increase of the uninformed search correlation indicates a preference for more thorough search of the robot's local area when searching without information. The CPFA turning angles increase from 12 to 19 degrees and the CPFAT increases from 5 to 9 degrees, as obstacles increase from zero to 1000. This indicates *i*) the GA consistently increases turning angles in robot movement when there are more obstacles, and *ii*) the addition of trails encourages use of straighter search paths. We hypothesize the increase in turning angles causes robots to search local areas more completely when there are more obstacles, and further, increased turning reduces repeated collisions with the same obstacle.

The major difference between the CPFA and CPFAT becomes apparent when examining the evolved pheromone following rate. The GA shows a strong preference for pheromone following when the CPFAT trails are used. As previously stated the trails are only utilized when a robot is traveling from the nest to a resource, not when returning to the nest. We hypothesize that use of trails in both directions would result in a significant increase in performance over this initial CPFAT implementation.

In the pathological case of the bug trap, the methods show distinct divergence in performance. Both methods use random movement to escape the trap. With the CPFA a robot that escapes and finds a resource cluster will return and either use memory or shared information, but both lead to a high probability the robot will become permanently trapped. In addition, a lack of shared information means that a single robot escaping the trap provides no benefit to other robots. The result is the noticeable increase in collision calls and decrease in fitness over CPFAT.

With the *Passage* and *Walls* arrangement fitness was decreased, similar to the trap, but

the difference between the CPFA and CPFAT are not as clear as the trap case. Unlike the trap, robots are less likely to become permanently trapped. Fitness decrease is more likely attributed to the difficulty of finding passages between regions and longer transit times between resource and nest.

Fig. 3.8 shows a detail of the simulation where a robot using CPFAT lays pheromone trails leading other robots to discovered resources. When a single robot escapes, finds resources and leaves a trail, it not only guarantees it's own escape again but other robots can escape using the same trail. The decrease in collision and increase in fitness are primarily due to a single robot succeeding and recruiting others via trails.

These results have another practical implication. Decreased collisions and increased fitness were most apparent when the bug trap was used. This suggests that random placement of obstacles rarely results in formation of actual traps, even when the environment becomes quite cluttered. Our expectation is that obstacle arrangements in real-world situations would more closely mirror the random distribution rather than the pathological case. While bug traps provide an important test of how different algorithms function under adversarial conditions, they may be rare in natural environments. Thus, robot swarms designed to operate under real-world conditions may not require use of complex and time consuming methods of path-planning or obstacle avoidance. A simple strategy coupled with use of multiple agents may be adequate to handle a majority of encountered conditions. Simple trail following algorithms with stochastic movement can be effective because only one robot needs to escape the trap and lay a trail which can be followed by other robots. Additionally, since swarms have multiple robots, the cost of having a few robots trapped is lower than it would be for a lone robot.

Video demonstration of results available: [1]

[1] https://www.youtube.com/watch?v=yl9YAl43u-Y

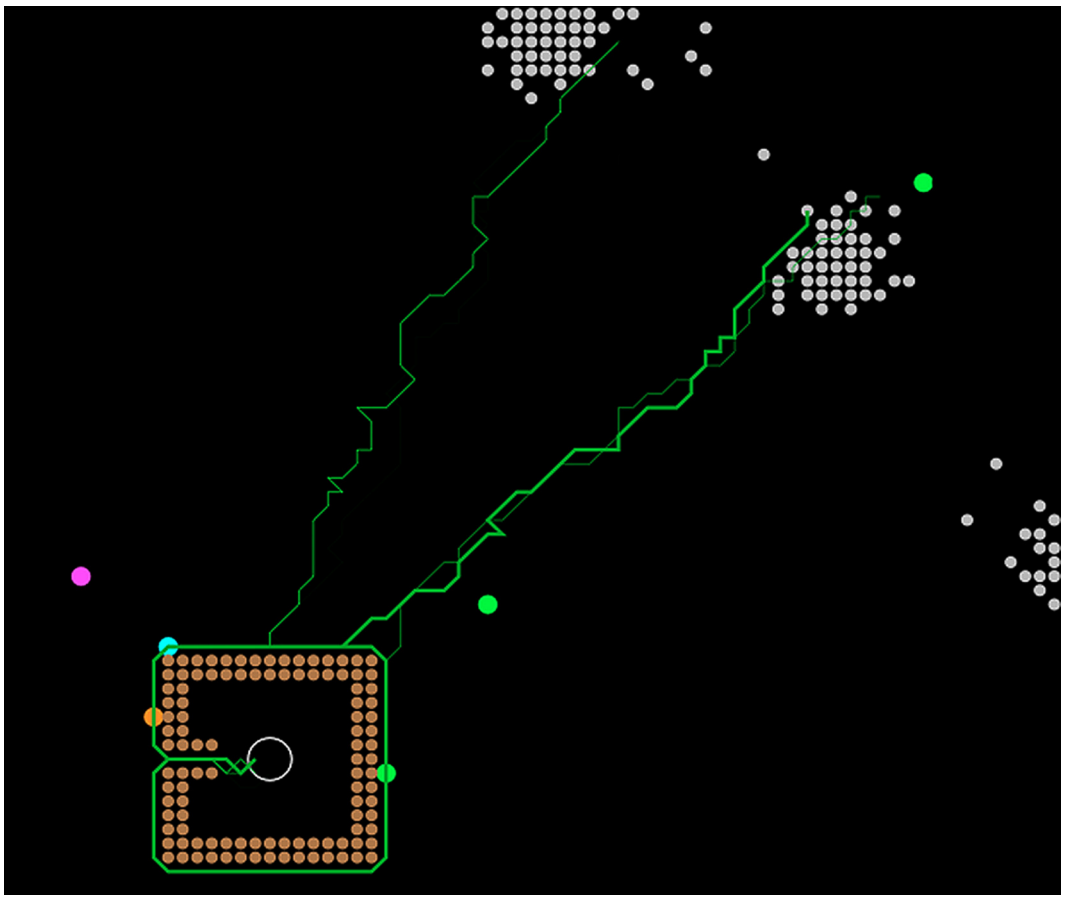


Figure 3.8: Simulation detail showing trap, pheromone trail usage and robot states: Searching (magenta), Returning sans resource (orange), Returning with resource (green) and Returning to site via trail (cyan). Nest is in center of trap.

3.7 Conclusion

We have demonstrated that a simple shared information strategy can allow a group of small, low-cost robots to successfully navigate an obstacle laden environment via evolution of existing behavioral parameters. Not only does this method work in a random distribution of obstacles but use of stygmergic trails allows success when faced with the classic bug trap. This simple method is a starting point for future research using more sophisticated obstacle avoidance strategies.

Chapter 4

Conclusion

This thesis only presents the beginnings of a successful robot platform and obstacle avoidance technique. Future work, already under way, includes adaptation of the localization method to use a cadiotropic mirror and the necessary physical adaptation to fit the mirror onto the robot. Work has also been performed to adapt the iAnt CPFA(T) to a different physical robot platform and use of the Robot Operating System (ROS) and Gazebo simulation environment while maintaining the iAnt's core principals. Our hypothesis has always been that the iAnt CPFA(T) should be adaptable to any system of terrestrial robots, no matter size or basic hardware configuration. Lastly, the bug-algorithm used for obstacle avoidance is one of the most simple available and this work is but a starting point for further research, rather than a comprehensive solution. More sophisticated avoidance algorithms that do not need global information can certainly be tested and in fact, should be.

Our hope is, like the natural evolution incorporated into the CPFA, the iAnt is a system which will continue to grow, evolve and succeed far into the future.

Appendices

A iAnt Version One Manual

B iAnt Version Four Technical Drawings

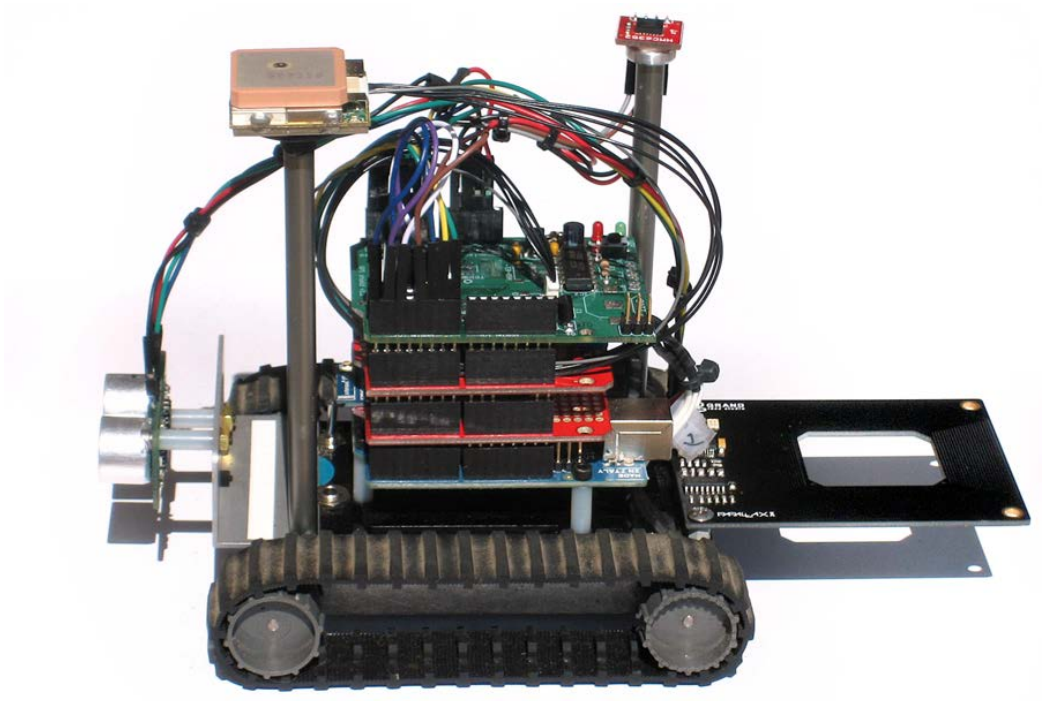
C iAnt Version Four Parts List

Appendix A

iAnt Version One Manual

The AntBot Project

SCALENET LAB - UNM



Owner's / User Manual

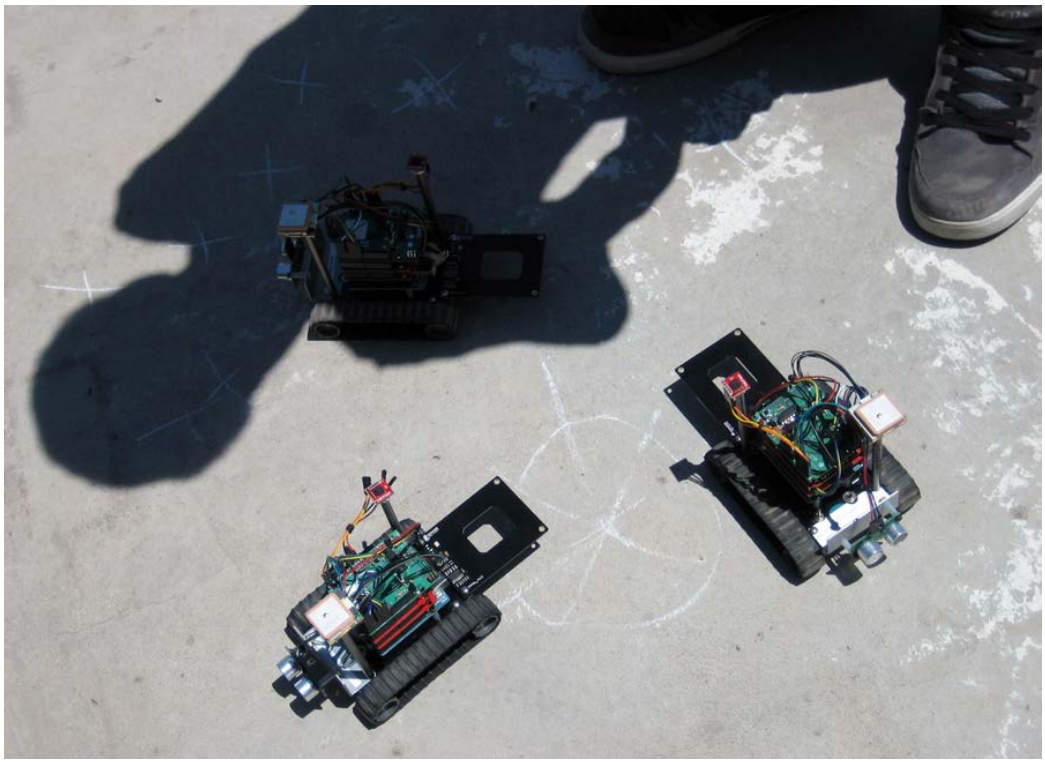
AntBot Version 1.0 – Table of Contents

AntBot Version 1.0 – Overview

The intent of this document is to provide a complete operations manual for the robots constructed for the physical demonstration of an ant-based genetic algorithm as overseen by Dr. Melanie Moses at the University of New Mexico SCALENET Lab.

This guide is intended for any end user of the "AntBot" or anyone seeking to build such a robot for research and development. Design considerations are included, as are original, manufacturers technical documents. Also included is current documentation (as of the date 7/23/11) for software libraries written for the AntBot for use with the Arduino IDE.

Team members for this project include PhD candidate Joshua Hecker, undergraduate Karl Stolleis (author) and high school intern Daniel Washington.



AntBot Version 1.0 – Hardware Description

Physical Platform

The OSBots.com Surveyor-1 chassis, Arduino version was chosen as the platform for the project. The case is machined aluminum and contains a 7.4V (2-cell) Lithium Polymer battery, four 100:1 gear-motors, a charging jack and on/off switch. The top plate is polycarbonate and has holes drilled and tapped (#4-40) for mounting the Arduino microcontroller board.

Primary drive is via two, rubber tank-style treads attached to the motors on each side. The tread drive wheels are driven directly by d-shaped motor axles.

In use the platform has shown to be light and sturdy. A maximum climbing angle, on a smooth surface, was approximately 30 degrees.

Microcontroller

The actual control unit for the AntBot is the Arduino Uno (smd version) microcontroller board based on the Atmel AtMega 328 digital chip. The board has provisions for 14 digital pins, 6 analog pins (routed through a DAC) and 6 of the 14 digital pins can be used for pulse-width modulation (PWM). There are onboard regulators to supply 3.3V and 5V power while taking 7-20V DC input. Programming is via a USB port on the board. One of the notable features of the Arduino system is the stacking header system that extends each of the motherboard's pins up through each attached daughter board allowing for electrical and physical connections between the boards. Care must be taken to avoid overlapping pin connections when using this system and without an additional power source the motherboard is only capable of delivering 750mA of current to the attached shields. Each shield will be detailed later in the document.

The Atmel AtMega 328 is an 8-bit native processor running on an onboard clock of 16Mhz. Total RAM storage on board is 32K with 0.8K being taken up by the boot loader. The processor is programmed via a USB bus using the Arduino IDE. All logic lines use a 5V reference voltage. Care must be taken to ensure that all attached devices are 5V logic tolerant or provision must be made to limit voltage lest damage occur to the attached device.

A deciding factor for the use of the Arduino is its Open Source Hardware design. Design information as well as basic programming examples are provided by the Arduino developers and available from multiple sources on the internet.

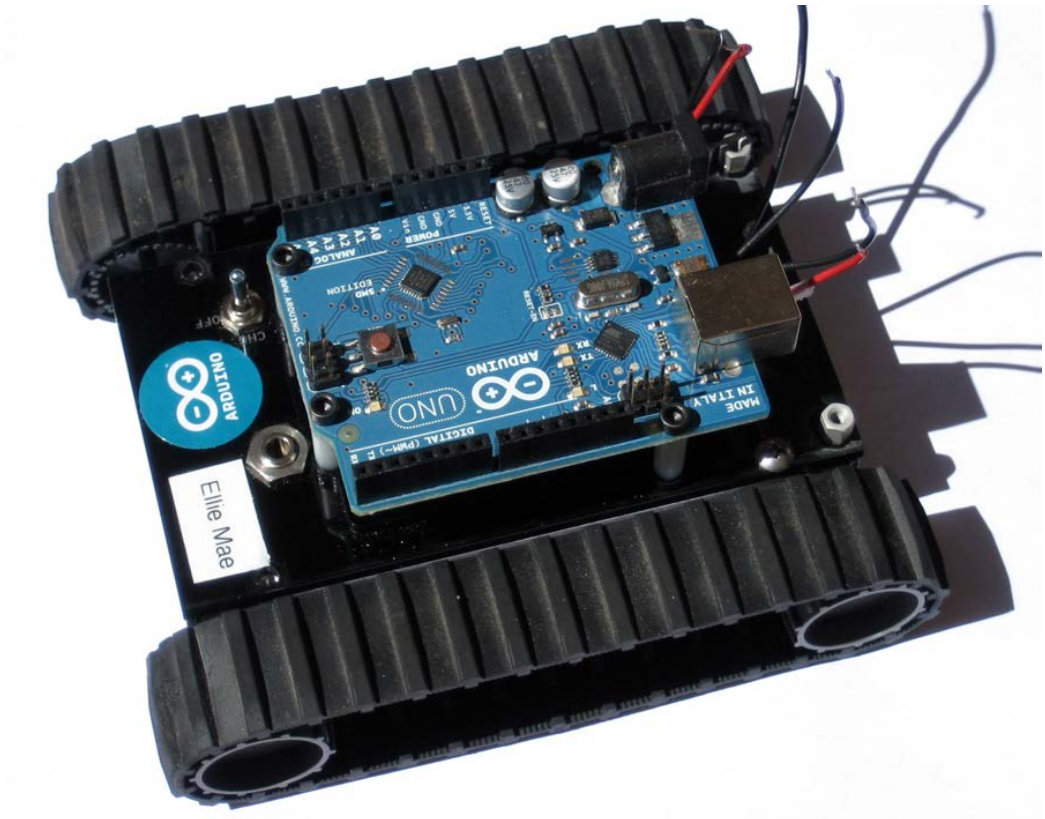


Figure 1 - The Surveyor Chassis with Arduino Attached. Standoffs can be seen underneath main board. Power switch and charging port are on left. Wires on right are for main power and motor connections.

Wireless Networking Shield

The first shield attached to the Arduino Uno is the Sparkfun, Inc "Wifly" board. This board uses the Arduino stacking header system for physical and electrical attachment. The shield utilizes the Roving Networks Wifly GSX 802.11 b/g wireless module. This module allows for normal 802.x communication with the robots over any accessible wireless network. The interface with the Arduino is via the microcontroller's SPI data bus which is tied to pins 11, 12 and 13 with a fourth pin required for device selection (the SPI bus allows for multiple device connections that all share pins 11-13). Range for the Wifly GSX is listed at 300+ ft but practical use has shown a reliable range of slightly less.

The Wifly unit also has provision for mounting an external 2.4gHz antenna via an onboard UFl connector. An adapter is needed to interface this with the more commonly available antennas (RPMSA) types and this connector is fragile so although this capability is available the

increase in physical complexity might outweigh any advantage gained from extended signal range.

The practical use of the Wifly system is to allow two-way communication with the robot via a local wireless network. The data collected from positional and environmental sensors are be transmitted via the network to a personal computer while commands and data sharing between robots are sent via this system.

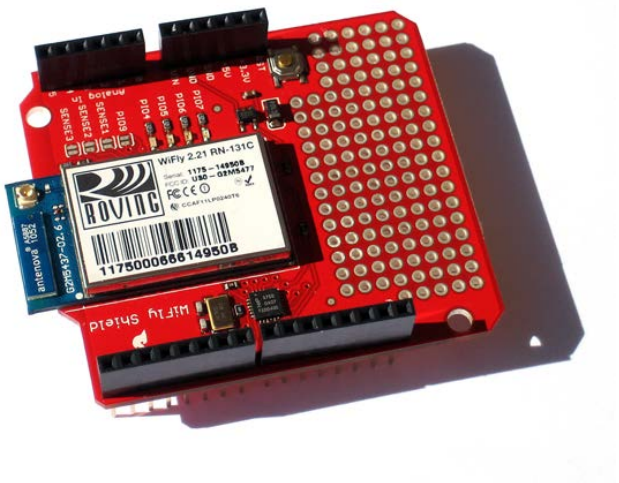


Figure 2 - The Roving Networks WiFly module on mating Arduino shield.

Motor Control Shield

The next shield attached above the Wifly board is the Sparkfun, Inc. "Ardumoto" board. This board uses an L298 Half-Bridge to supply two separate control lines with a maximum current draw of 2 Amps. Onboard power connections allows motor voltage inputs of 6-18V and also supplies the Arduino with power via header connections. This is the method chosen to deliver power to the system since higher current draw for the system could exceed the capabilities of the Arduino with all four motors running. In practice this system seemed to provide necessary current to all parts without damage or failure.

The Ardumoto board has one drawback related to its basic design. The board uses pins 3, 11, 12 and 13 causing a conflict with the SPI bus for the aforementioned Wifly board. The solution for the initial runs of robots was to use jumpers to move pins 11-13 to pins 5-7 in the same order. This solution did cause problems with an apparent floating ground on the header containing pins 8-13 so arrangements were made to keep those pins unconnected. For future robots the suggestion would be made to use the motor driver board made by DFRobots. This

board uses the same half-bridge and therefore the same software commands while pin connections are made through pins 5, 6, 7 and 8. This solution would seem to eliminate the need for jumpers and would free two additional digital pins for possible connections/usage.

It should be noted that there is no marking on the board for motor connection polarity only note of right and left side. Some experimentation may be required to connect the motors properly to allow for expected results.

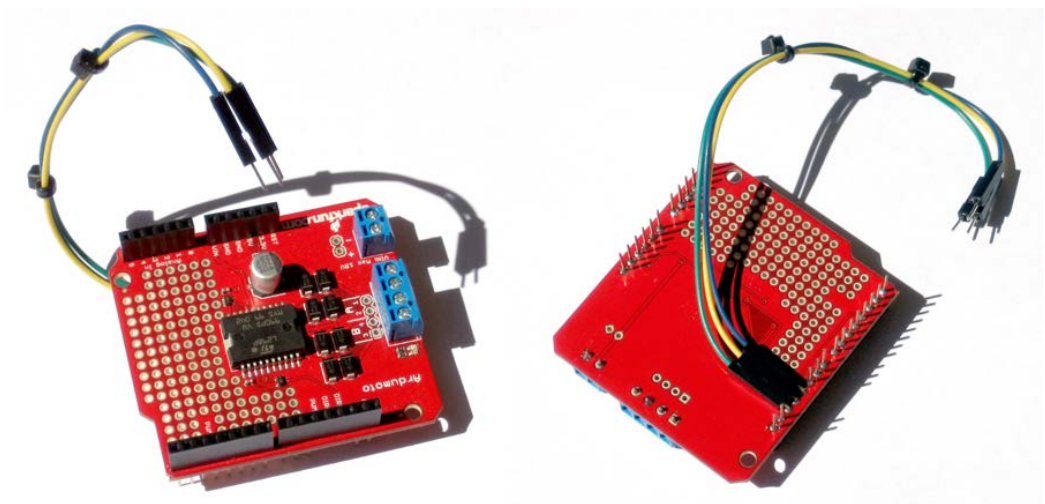


Figure 3 - The Ardumoto Shield with attached jumpers (top and bottom). The jumpers are the three pins that need relocation due to conflict with wireless board. Blue screw terminals (right) are for power and motor connections. Perforated area on left is prototyping area.

GPS/SD Shield

The final board on the electronic stack is the GPS-SD data-logging shield manufactured by Adafruit Industries. The board has provision for mounting a GPS receiver that uses a six-pin FTDI connector, a coin-style battery for power backup, a FAT formatted SD card and two led's for indicators. The Arduino Uno's main reset button is also brought up to this shield. Physical and electrical connection is made through the Arduino's stacking header system.

A logical choice for a GPS receiver on the Adafruit board is the EM-406A from US GlobalSat. Further discussion of the 406A will be made later but some basic considerations will be discussed here. The Adafruit shield and the Arduino supply 5V logic lines which are native to the 406A but many GPS receivers operate on 3.3V logic voltages necessitating the use of an external step-down system and complicating physical design of a small robot such as the AntBot.

The GPS shield provides a mounting area for the GPS receiver on the board and adhesive foam-backed tape was used to mount the receiver. With this arrangement it was noted that

excessive lock times for the GPS were present even when using the unit outdoors with a clear view of the sky. Experimentation demonstrated that removing the receiver from the board and mounting it just two inches away, atop an aluminum stalk, provided lock times corresponding to the 406A technical specifications and was able to maintain a data lock even indoors. The final design of this series of robot included replacing one of the mounting screws for the polycarbonate top plate with an aluminum stalk approximately 10cm in height and a longer FTDI cable was used to connect the receiver with the port on the GPS board.

Another consequence of using the Sparkfun, Inc Ardumoto board and the necessary re-routing of pins 11-13 is that the SD card read/write capabilities of the GPS shield could not be utilized. SD card connection is made through the hardware SPI bus and the rerouting of the motor board pins caused a failure of this connection. The previously mentioned alternative motor board from DFRobot would, in theory, allow for use of the SD card feature on the GPS board.

After relocating the GPS receiver to the external stalk consideration was given to removal of the GPS shield altogether but ultimately the board was left in place for several reasons. GPS receiver connection was much simpler made through the board and the Arduino reset button were considered important enough to continue use of the board. Also, if in the future the SD card feature could be utilized with the possible motor control board substitution.

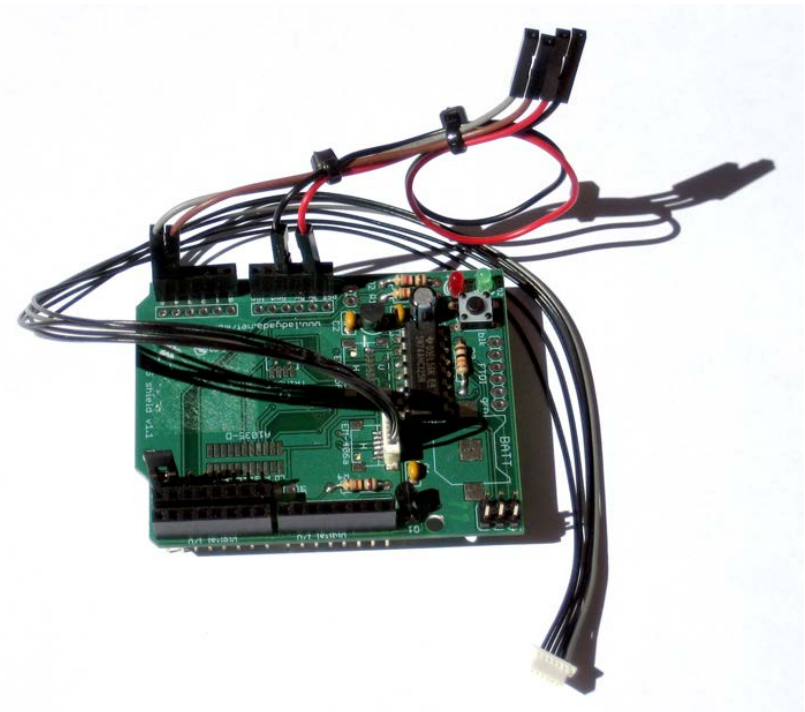


Figure 4 - GPS Shield with FTDI cable and jumpers for compass attached. Shield also has main reset button (upper right) and led's for indicator use.

GPS Receiver

The choice of GPS receiver is the EM-406A from US GlobalSat. This receiver interfaces easily with the Adafruit GPS shield, via a six-pin FTDI cable as well as being compatible with the Arduino 5V logic levels. The unit is small, lightweight and is listed as having a WAAS-enabled accuracy of 5 meters. In practice we found this receiver to have slightly better horizontal resolution when used with a clear view of the sky. The receiver also uses NEMA standard 0183. The NEMA standard data consists of comma-delimited strings that are easily parsed within the Arduino framework. Use of this standard also makes substitution of another GPS receiver easy while using the same software.

The 406A also uses a data update rate of 1Hz limiting access to additional readings and possibly limiting the accuracy of positional data. In the future GPS receivers with higher update rates might be considered for use but high volume of data contained in the NEMA standard may cause processing delays with the Arduino.

Initially connection of the GPS unit was made through digital pins 2 and 4 requiring the use of a software library (NewSoftSerial) to mimic hardware serial communication through digital pins. Eventually the GPS unit transmit and receive lines were switched to the Arduino pins 0 and 1. These pins are dedicated to hardware serial connection with the Atmega 328 and produced faster and more reliable data acquisition. The NewSoftSerial library was not necessary for this connection. The apparent speed increase was due to not having to use software to mimic the serial ports on standard digital pins. The one minor complication in using pins 0-1 was that the connection to the GPS unit must be removed during programming of the Arduino via the USB cable, due to the fact that the programming takes place on the same pins. Once programmed the GPS connection is re-established and showed no ill effects during normal operation.

The GPS unit serial communication is also at 4800 baud so care must be taken to initialize the Arduino's baud rate to 4800 and also to use 4800 baud when monitoring the debugging GUI connected to the serial port. These issues were minor when weighed against the advantage of stable, reliable data communication with the GPS receiver.

The function of the GPS receiver, on the robots, is positional location, route tracking, waypoint information (home and "food" location) and generation of paths to and from waypoints for distribution to other "ants." The limitation of the GPS unit is the requirement of a reasonable unobstructed view of the sky and the need for the unit to be moving to generate directional heading information.

Figure 5 - EM 406A GPS Module attached to aluminum stalk. The threaded end replaces top-plate chassis screw.

Digital Compass Module

The unit chosen for external compass readings is the Honeywell 6352 digital magnetic compass. The compass module used was manufactured by Sparkfun, Inc and allows for communication over the Arduino's native I2C bus. The I2C protocol is an industry standard that allows for communication with I2C devices via two wires (SLA and SLC) tied to the Arduino's analog pins of A4 and A5. Up to 127 devices may be chained together on this bus with communication made possible via a slave/master command sequence transmitted via the software.

The compass module is 5V logic tolerant and able to run on VCC of 2.7-5.2V so power connection was made via jumper to the Arduino's 3.3V regulated supply pin. The SLC and SLA pins were connected via jumpers to their respective pins on the Arduino.

The compass was originally mounted on an unused area of the GPS shield but moved to an aluminum stalk, similar to that of the GPS receiver, replacing a screw holding the polycarbonate top plate on the chassis. The stalk is a 7075 aluminum tube with aluminum insert, on top, to hold the compass module. The insert was a sliding, interference fit to allow for alignment of the compass module, with respect to the robot chassis, while preventing the compass module from inadvertently rotating during routine use. The decision to use the compass atop the aluminum stalk was made in light of the compass module's apparent sensitivity to electromagnetic fields generated by the chassis motors. The height of 10cm was chosen to help maximize the distance between the compass and the motors, as well as any other possible EMF sources, while maintaining a balance to the overall size of the robot.

The need for the compass module is to allow for directional information to be gathered while stationary. During periods of little or no linear motion the GPS unit cannot be relied on for

reliable position information other than latitude and longitude. The eventual desire is to correlate the data from all positional sensors and allow for greater location accuracy than any one sensor can provide.

Figure 6 - HMC6352 Compass module on aluminum stalk. Threaded end replaces chassis top-plate mounting screw. Mounting plate is slip-fit in tube to allow alignment.

Ultrasonic Rangefinder

The need for distance measurement and obstacle avoidance on the robot is handled via the Devantech SRF05 Ultrasonic Rangefinder unit. This unit uses two digital pins on the Arduino for communication and is connected to the regulated 5V and GND connections on the Arduino via jumpers. The effective range of the unit is 4 meters, which in practice allows for positional location within the GPS data's error margin of approximately 3 meters. The unit allows for updates every 10 microseconds and can be configured to operate over a single communication jumper, should additional digital pins need to be freed up on the Arduino.

The SRF05 was originally mounted on the front of the Surveyor chassis but initial testing showed sensitivity to surface irregularities, such as expansion joints in concrete, providing false obstacle readings. This failure was attributed to the units close proximity to the travelling surface and the unit was eventually raised by mounting it on a 1"x1" aluminum angle attached to the front of the robot. The polycarbonate chassis mounting screws were again utilized to minimize the modification to the original chassis.

Figure 7 - SRF05 Ultrasound with mounting bracket. Notches in bracket, near mounting holes, are relief for power switch and charging port.

RFID Read/Write Module

The final piece of electronic hardware, on the robot, is the Parallax RFID Read/Write module. This module allows for use with 125kHz RFID tags for both reading and writing data on the tags. The tags used consist of a flat, plastic card with an RFID chip imbedded in the card. The chip is passively read, via the read function, and contains a unique serial number assigned by the manufacturer that allows for consistent identification of each tag. In addition to the read-only feature the tags allow for the writing of 116K of information in 31 register locations on the RFID chip. The manufacturer lists the interaction distance at 3 inches under optimal condition

The initial design was to install the RFID modules with adhesive on the underside of the Surveyor chassis to keep the robot size small and the reader unobtrusive. It became apparent that EMF interference from the chassis motors was providing false readings and unstable data connections with the RFID reader. Concerns about physical abrasion of the module, while mounted under the chassis were raised but data reliability decided the final placement of the unit. The final configuration uses small, threaded, aluminum standoffs glued to the rear of the polycarbonate top plate and allowing for attachment of the module via 4-40 screws. Despite the increase in physical distance from the card the data connection reliability rose to a reasonable level when the RFID card is immediately under the read/write module.

The RFID module data connection is a hardware serial connection that is made either via the hardware serial connection on the Arduino or via any other two digital pins using the NewSoftSerial library to mimic serial operation of those pins. As the GPS receiver occupied the hardware serial port it was necessary to choose digital pins and use the software serial solution.

The unit works reliably under this system, more reliably than the GPS receiver, due most likely to the much smaller volume of data being transmitted and received via this interface.

The power needs of the module are 5V regulated and with the Arduino only having one regulated 5V port available via headers a "wye" jumper was made to allow simultaneous connection of the RFID module and the ultrasonic rangefinder to the single 5V and GND ports.

The use for the RFID tags is to simulate the robots need to locate and pick up "food" in the same manner that ants collect food from seed piles in the wild. The cards can be located, serial number scanned for "food" location identification and the size or number of seeds in the "food pile" can be set and decremented by the robots thus simulating an ant retrieving a seed from a pile in the physical world.

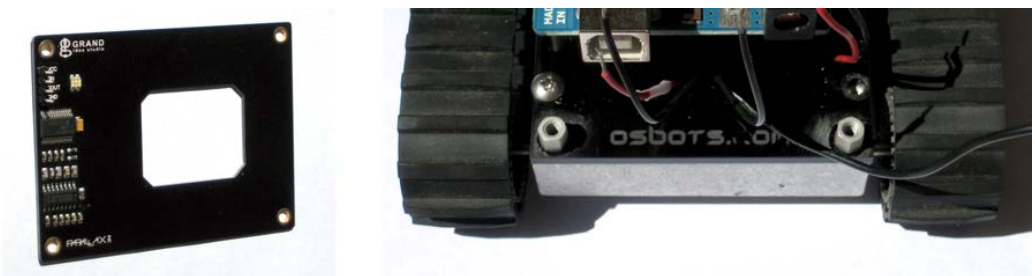


Figure 8 - RFID Module and rear of chassis showing mounting standoffs. Header pins, as supplied were right angle and changed to straight.

Hardware Assessment

The only real testing done with the hardware, once in final configuration, was to test current draw and ensure that the provided battery was capable of powering the system. The robot was powered via a BK Precision digital power supply and all modules/sensors were activated and motors were run at a speed setting of 200 out of a possible 255. The noted free-running current draw was 0.69 amperes and when running with a load of only its own weight the current draw was .0.75 amperes. When the motors alone were used the current draw was 0.5 amps which places the system within the current supply capabilities of the onboard 7.4V battery and the current handling of the Arduino Uno.

Additional Information

For additional information on each of these hardware items consult the attached manufacturers technical documentation in the appendices of this manual.

Information on the Arduino Uno environment, hardware and IDE can be found at www.arduino.cc. This site contains information on the physical hardware, included software libraries, third-party libraries, available hardware shields and a user forum.

AntBot Version 1.0 – Maintenance and Known Physical Issues

Physical Maintenance

The Surveyor chassis was established to be quite rugged under normal use. Some maintenance issues did arise during use.

Due to the close proximity of the axles to the ground it became apparent that debris was easily collected by the axles and could contribute to a substantial drag on these small motors. The nature of the debris was mostly carpet fibers, pet hair, string, ect.. This issue requires periodic checking and cleaning of the motor axles. Removal of the treads and drive wheels allows for easy access to the area.

Drive wheel removal is accomplished by sliding the wheel straight off the d-shaped axle extending from the motor. The wheels are pressed on with reasonable hand pressure and care should be taken to pull the wheel straight off to avoid damage to the motor gearbox.

Lubrication of the gearboxes is probably not necessary but due to the relatively open nature of the gearbox and proximity to the ground a dry, powdered graphite would be a good choice, rather than oil.

The motors seem durable in use but a fall from a table showed that the gearbox was susceptible to failure. Any user of this chassis would be advised to have at least one spare motor on hand to limit operational downtime. Replacement of the motors requires removal of the top plate, one screw and the use of a soldering iron.

The included battery on the Surveyor is a 7.4V 2-cell Lithium Polymer battery with no provision for cell balancing during charging operations. Care must be used to charge these batteries with only the supplied charger or with a charger specifically designed for LiPo batteries. If any other type of charger is used the batteries may be damaged and can burst due to overheating or even erupt into flames. Additional consideration might be given to not leaving the robots unattended during the charging operation due to damage concerns.

The charging is via a small port on the front of the robot and the on/off switch must be set to the charge position. The charger has an LED that indicates a charging condition (RED) and a full charge (GREEN).

All screws on the chassis are 4-40NC threads and are interchangeable. Any modifications to the initial robots were made with this in mind and all additional mounts kept this thread specification.

Also the use of adhesive backed foam was used for the mounting of any sensors, other than the RFID unit, to allow for easy removal or relocation. The use of any more permanent adhesives is discouraged unless the final physical layout is well established.

Little other maintenance is required of these robots using the system outlined. Of note is the availability of round wheels for the Surveyor chassis. The wheels are available at additional cost from the manufacturer and might provide a more stable sensor surface if operating exclusively on smooth surfaces. Finally assembly of these robots, although minimal, does require the use of a soldering iron and a multimeter would be recommended for testing and troubleshooting.

AntBot Version 1.0 – Operation and Known Issues

Honeywell 6352 Compass Module

The 6352 module is a small, light and inexpensive compass module used on the AntBots. Its unique attribute, among digital magnetometers, is the use of an onboard processor to provide compass headings in degrees. Stated accuracy of the compass is 2.5 degrees. Other comparable units are magnetometers only therefore the main microcontroller must be utilized to interpret the data and translate the readings into useable headings. This unit is not tilt-compensated so a level and stable operating surface must be maintained to give accurate readings.

Initial testing consisted of out and return tests and programming the robot to map out a square pattern based on the four cardinal directions, with side lengths of one meter. The compass unit is factory calibrated but initial results contained approximate angular error of +/- 15 degrees from known cardinal headings. The reference for the established headings was via a Suunto KB-20 handheld compass. The compass has provision for user calibrations and a function for the project Compass.h library was written and after calibration results were improved but showed errors of +10 degrees on the headings of 0 and 180 degrees. Since the compass responds to the earth's magnetic poles this discrepancy could not be attributed solely to magnetic declination.

Additional attempts were made to access and alter the four register locations containing the calibration offsets for the compass module. These locations were accessed via additional functions in the Compass.h library and were successfully overwritten. On the initial unit this proved helpful but when the same correction offsets were loaded into subsequent units erratic and varied results were obtained. The conclusion of the calibration testing was that the internal calibration routine was not ideal but without direct parsing of the raw magnetometer data further improvement of the user calibration routine could not be easily accomplished.

Care was taken to calibrate the compass modules of all robots under identical conditions and with some care to remove sources of magnetic interference, such as large ferrous objects. A small turntable was assembled to facilitate the level and smooth rotation of the robot during the compass calibration.

As noted in the physical hardware descriptions the use of an external standoff was used to minimize any EMF interference with the compass from either the chassis motors or other electronic components. It was noted, anecdotally, that large ferrous objects such as metal support poles could influence the compass' data when in proximity of 1-2 meters.

Although initial results with this compass were disappointing when compared to headings obtained with the GPS it is obvious that the compass cannot be dispensed with altogether and all due diligence must be made to maintain accuracy of this unit.

EM-406A GPS Receiver

Data from the GPS receiver, as previously noted, is formatted using the NEMA 0183 standard, common to many GPS receivers. Relevant data, to this application, was determined to be date, time, latitude, longitude, heading, speed, horizontal dilution of precision and number of satellites used in the calculations. Each of the raw data streams is parsed via the project's GPS.h library. A C-style struct is used to store the data internally and return the above fields to the main program. All are returned in the string format except for latitude and longitude. Although checksums are included in each of the raw data strings they were not utilized in this version of the project for error checking.

Additional processing of the latitude and longitude is needed to ease calculations with this data. The original format is a string of the "degrees, minutes" form (ddmm.mmmm) with a precision of four decimal places for the minutes. This was converted to decimal degrees (dd.ddddddd) and seven decimal places were used as the precision due to the conversion of essentially base 6 notation (minutes) to base 10. (60^4 = 1296000 so precision of 10^7 places). These data fields are returned as floating point decimals to facilitate mathematical calculations using latitude and longitude.

Note must be made of the horizontal dilution of precision (HDOP) field. This number ranges from 1.0 to infinity and is effectively a multiplier for the known error of the system. Established error for WAAS corrected GPS data is 3m therefore an HDOP value of 1.0 yields an error of 3m, the theoretical maximum of terrestrial GPS.

Initial testing with the unit showed stationary accuracy of approximately 3-5 meters. Additional testing, while the robots traced out a 10, 5 and 3 m square showed useable resolution down to the 3 meter range.

As noted in the hardware description a change was made to the initial configuration to relocate the GPS receiver atop a stalk similar to that used with the compass module. This created a noticeable improvement in unit position lock time and stability. No testing, aside from observation was used in this decision. It should also be noted that the led on the receiver used to indicate a position lock (blinking) was a reasonable indicator of valid data from the unit. Although position, date and time data could be read from the unit without an indicated lock it was

intermittent and functionally inaccurate with the exception of date and time. Most often when a lock was not indicated polling data from the unit results in no data being read at all.

As was also noted this unit is connected via the hardware serial ports of the Arduino (pins 0,1). The initial configuration, using other digital output pins requires the use of a software library, NewSoftSerial.h , which mimics the function of the dedicated hardware serial on alternate pins. This arrangement is beneficial when multiple serial devices are used and is use on this project to connect the RFID module. The initial reason for relocating the connection was to resolve pin conflicts arising from the motor and Wifly shield but an increase in speed and data stability was noted leading to the conclusion that this is the ideal arrangement.

Note must be made of the ever increasing accuracy and availability of more accurate GPS units that could replace the 406A. Consideration must be given to number of channels available for tracking and the frequency of the update rate. The 406A has 20 available channels and an update rate of 1Hz. Units are already available with 60+ channels and 5-10Hz update rates but come with the caveat of increased power consumption and the ability to overwhelm the processing capabilities of many microprocessors when polled at a higher rate than 5Hz.

Devantech SRF05 Ultrasonic Rangefinder

Operation of the SRF05 rangefinder, in use, was quite simple and proved to be accurate out to the manufacturer specified range of 4 meters. A simple library, Ultrasound.h was written by the team and data returned in centimeters. Initial testing of the units showed accuracy to +/- 1 cm but further testing is necessary to fully establish the capabilities of this unit.
Concern was expressed about the AntBot's ability to not interfere with one another's ultrasonic readings. A simple test, with units approximately 20cm apart showed that with a delay of less than 10 ms that overlap of ultrasonic pulses was common, while the units were pointed directly at one another. A delay of 20 ms showed little if any error in the range detection.

Function was also included in the Ultrasound.h library for operation of the unit using a single wire to transmit and receive data. This is not necessary in the current configuration but might be necessary to implement in future configurations if additional digital pins need to be utilized on the Arduino microcontroller.

Parallax RFID Read/Write Module

This module was originally designed for use with the PIC-based Parallax microcontrollers. Little information was given for attaching this unit to an Arduino so a team written library, RFID.h, is used for communication with this unit. This unit will only work with

RFID tags containing the EM Microelectronics EM4x50 1kbit R/W transponder. Several versions are available. See the attached technical document for further explanation.

Testing showed that tag data could be read when approximately 40-50% of the tag is located directly beneath the reader. A slight improvement in read reliability was also noted when the tag's label was facing away from the reader. In terms of the ant-based model these robots are designed to replicate, being approximately one body length away from a food item is necessary for food detection

The library provides provision for reading the tag's unique serial number and any data location on the tag. The error codes provided by the tag are used to create a parity check of data coming from the tag and the reliability of the data received. Due to the passive nature of the tag activation and data transmission data errors are not unexpected and required some mitigation.

Read times from tag data are fast and despite a small delay in writing neither time delay is a noticeable in normal use.

Roving Networks Wifly GSX 802.11 b/g wireless LAN Module

Wireless communication with the AntBot is established with the Roving Networks Wifly module. The Wifly module is utilized via an Arduino compatible shield manufactured by Sparkfun, Inc. The board provides necessary power, logic and physical mounting on the Arduino Uno and communicates with the microcontroller via the dedicated hardware SPI serial communication bus.

Software dialog with the unit is via an Arduino third-party library Wireless.h. The signal from the robot is routed through a standard IEEE 802 compliant wireless router and has provision for signing onto a WEP protected wireless network.

Early testing of the wireless capabilities, to transmit sensor data from the robot to a laptop computer, show expected range capabilities and issues not altogether different from any wireless network. Few, if any other options exist, at this time, for wireless communication over standard wireless networks.

Sparkfun, Inc ArduMoto L298 Half-Bridge Motor Driver Board

The AntBot system is classed as a "differential drive" robot. This means that motors on either side of the robot work in concert to perform turns, rotations and movements without the use of any mechanical gearboxes or transfer systems. This requires the use of a "half-bridge" circuit to facilitate reversing the polarity connection to one pair of motors while leaving the second pair to run in normal polarity. One of the most common half-bridge circuits available as an integrated

circuit is the L298. This chip is utilized on the Sparkfun Ardumoto shield designed to work specifically with the Arduino Uno.

This board, via the L298 is capable of handling up to 2 amperes of current total. Testing on a BK Precision power supply showed total current draw of all four motors in the .5-.6A range. This is well within the capabilities of this chip without use of an external heat sink or any other heat management provision.

Note was made of the pin conflict inherent in this board, as designed and the recommendation to use the DFRobot Industries motor driver board for future iterations. This board also uses the L298 and should fit in with the other components and software with a minimum of modifications.

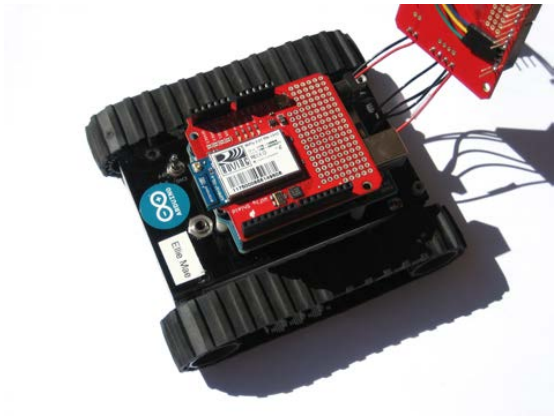


Figure 9 - Wifly Card on Arduino (stack 1)

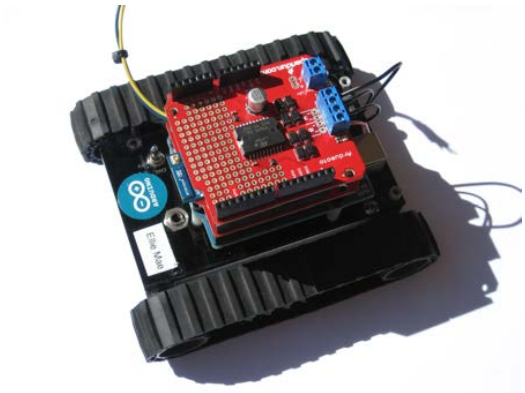


Figure 10 - Motor Driver Shield (stack 2)

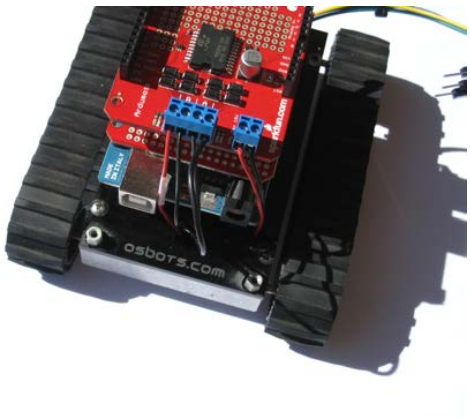


Figure 11 - Back of motor board showing connection

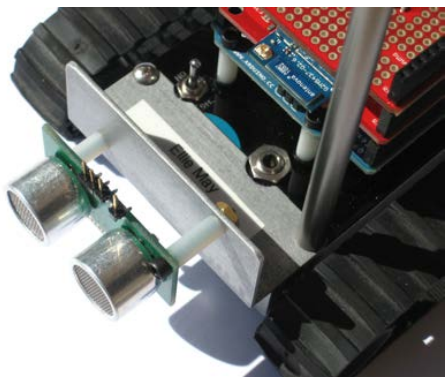


Figure 12 - Ultrasound Bracket using GPS stalk as one screw

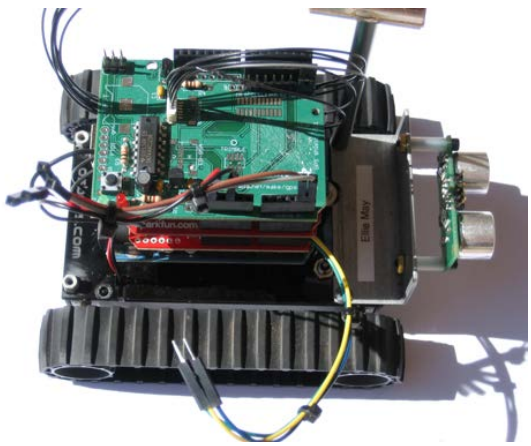


Figure 314 - GPS shield in place (stack 3)

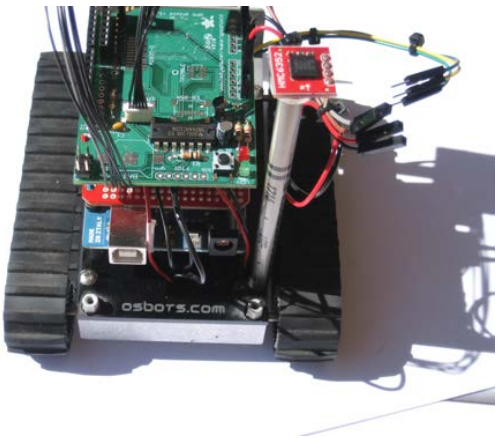


Figure 415 - Compass stalk in place

Figure 516 - Wiring harnesses

AntBot Version 1.0 – Arduino IDE Use

The Ardiuino IDE is used as the programming platform for the AntBot. The IDE has extensive documentation available at www.arduino.cc for included libraries and basic functions. The section immediately following is documentation of the libraries that were specifically written for this project. Additional third-party, external libraries were used and must be included for proper operation. These include : "NewSoftSerial," "Wire," "Random," and "Wireless." All can be found on the AntBot website as well as via the Arduino website. These libraries are used in their original form and were issued under the same creative commons license that governs the Arduino IDE. Since these libraries were not altered during this project they are not included in the software documentation that follows.

AntBot Version 1.0 – Part Sources

Arduino Uno SMD - http://www.sparkfun.com/products/10356	$29.95
Ardumoto - http://www.sparkfun.com/products/9815	$24.95
(Suggested Substitute - http://www.robotshop.com/dfrobot-arduino-compatible-motor-shield-2a.html $16.67)	
Wifly - http://www.sparkfun.com/products/9954	$89.95
EM-406A GPS Module - http://www.sparkfun.com/products/465	$59.95
HMC6352 Compass Module - http://www.sparkfun.com/products/7915	$34.95
GPS Shield - http://www.robotshop.com/adafruit-gps-logger-shield-kit-arduino.html	$19.50
RFID W/R - http://www.robotshop.com/parallax-serial-rfid-reader-writer-module.html	$49.99
Robot Chassis - http://osbots.com/shop/product/arduino-osbase/	$139.00
Total (version one)	$448.24

Appendix B

iAnt Version Four Technical Drawings

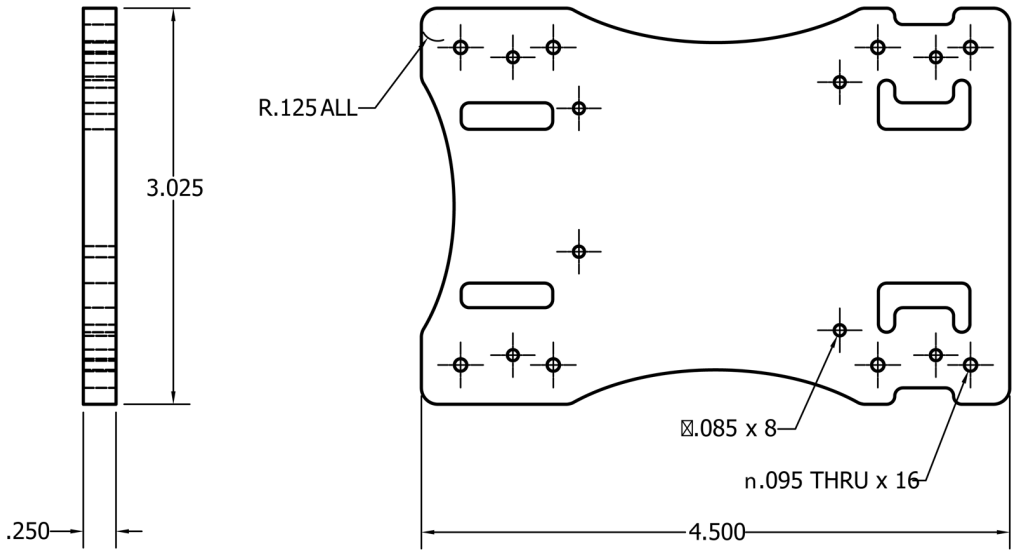


Figure B.1: iAnt Motor Base - 1/4” Laser Cut Acrylic

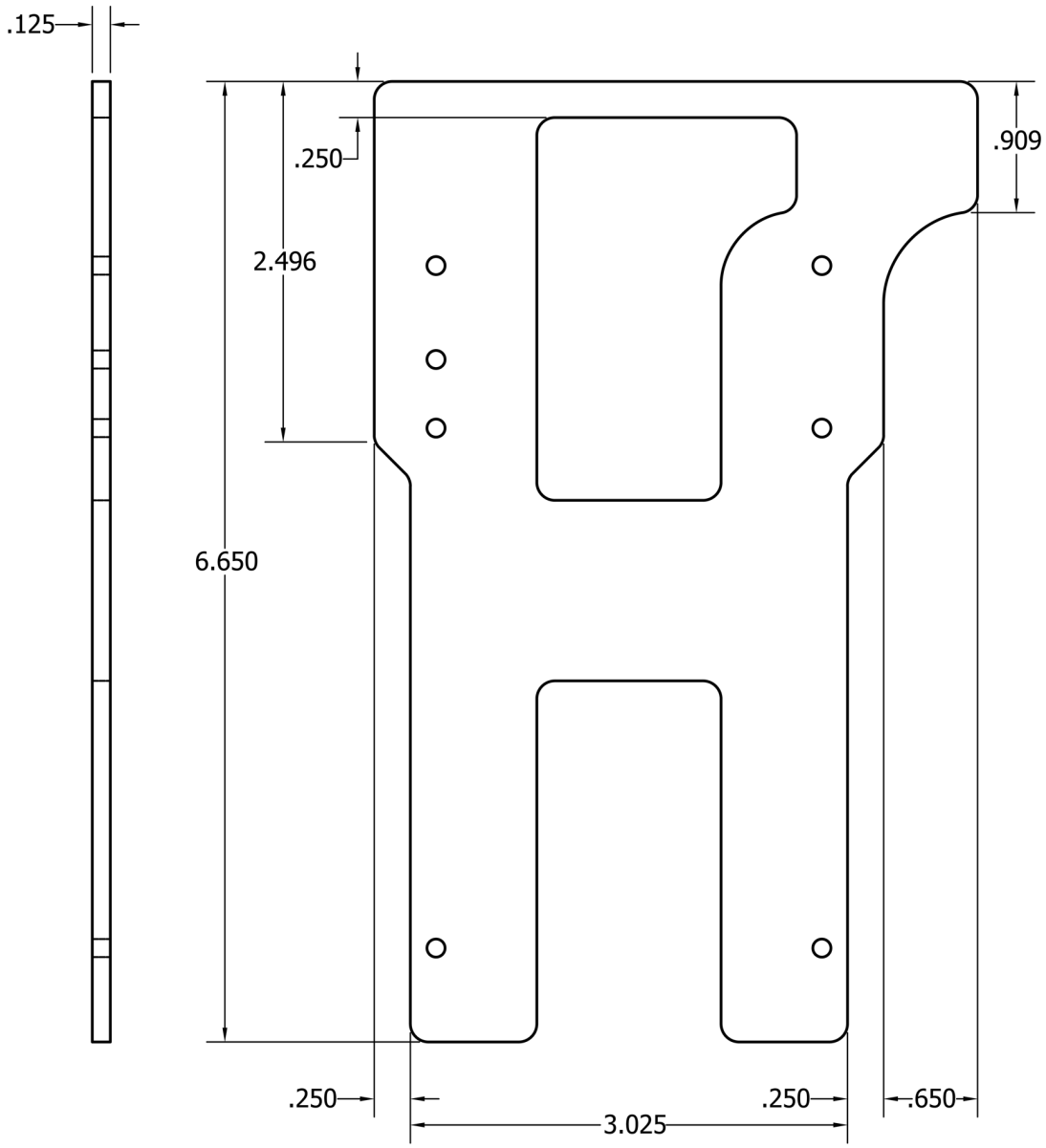


Figure B.2: iAnt iPod Base - 1/8” Laser Cut Acrylic

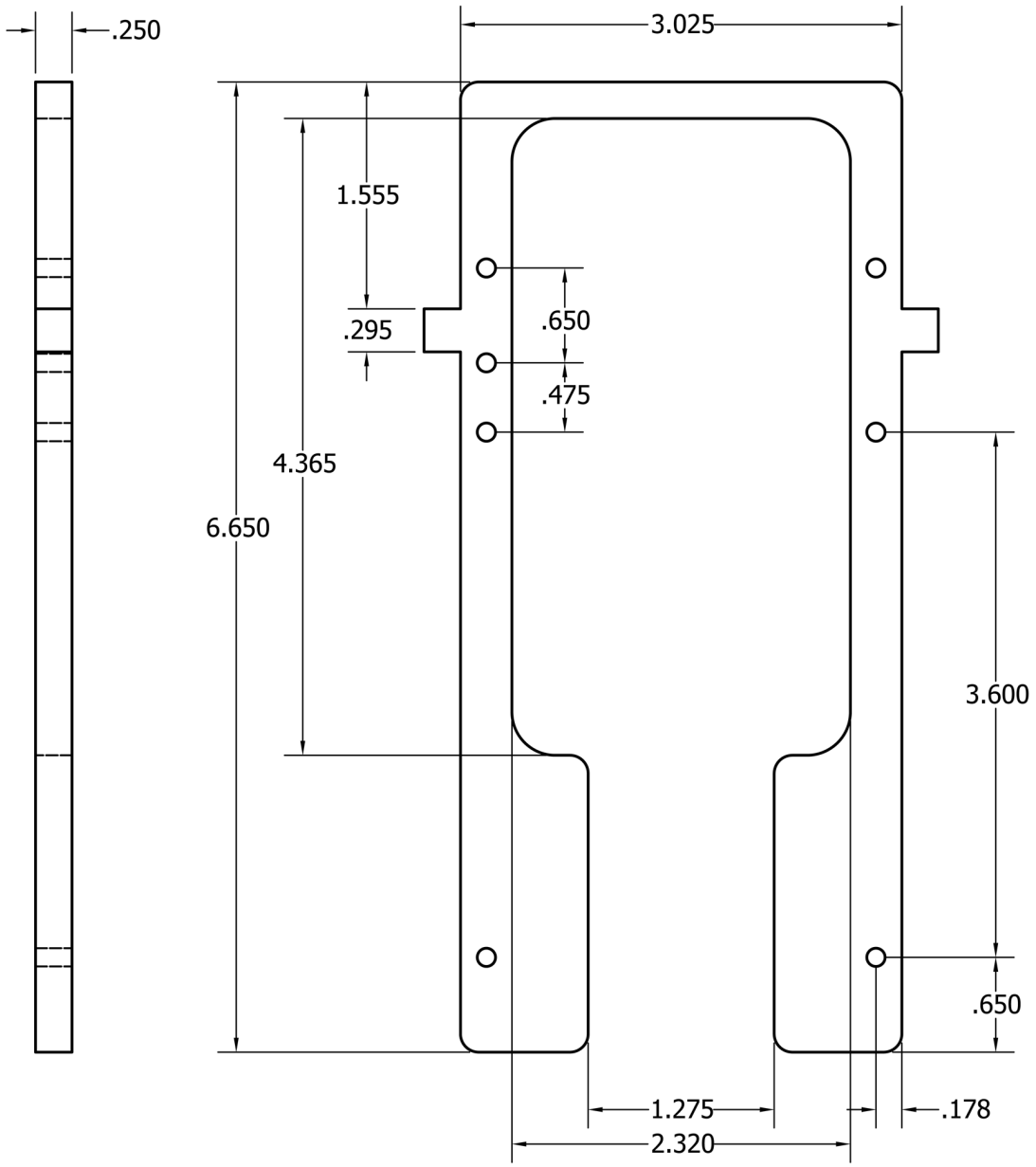


Figure B.3: iAnt iPod Cradle - 1/4” Laser Cut Acrylic

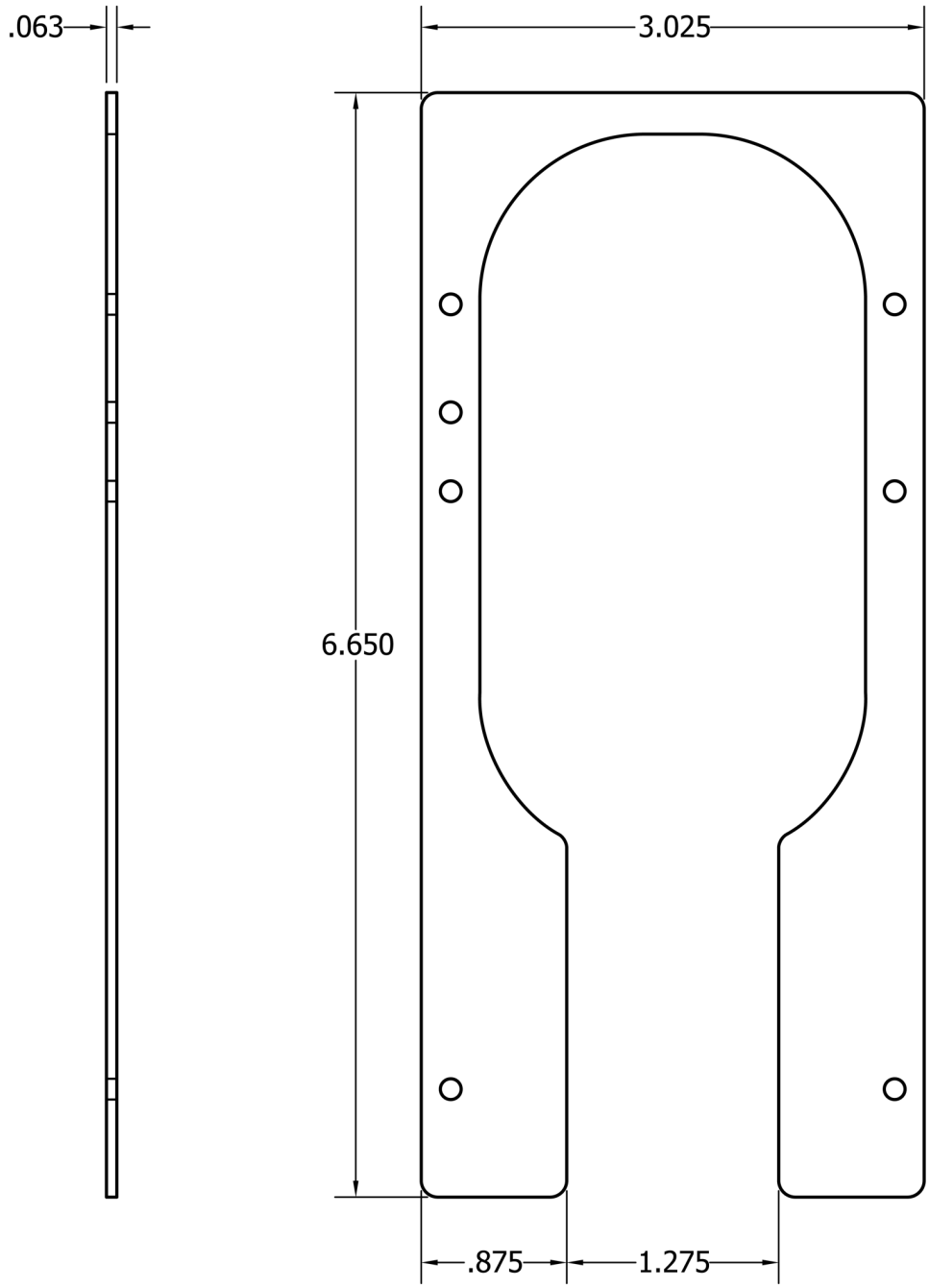


Figure B.4: iAnt iPod Lid - 1/16” Laser Cut Acrylic

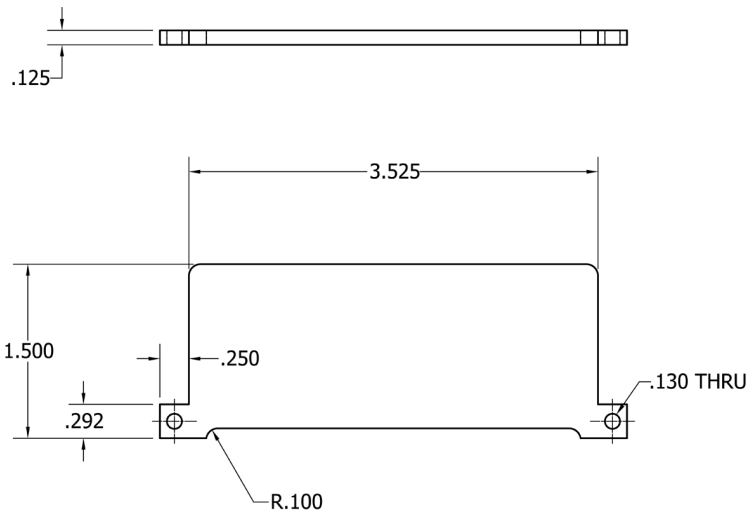


Figure B.5: iAnt Mirror - 1/8” Laser Cut Acrylic (Mirrored)

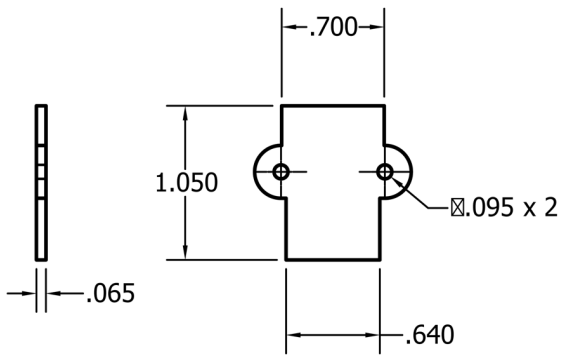


Figure B.6: iAnt Motor Cover - 1/16” Laser Cut Acrylic

Appendix C

iAnt Version Four Parts List

iAnt Master Parts List – Version 5 (2014/1/15)

Description	Manufacturer	Mfg Part #	Price Per Unit	# Needed	Source	Cost	Notes
Chassis – Wheels (supplied in pairs)	Pololu	1090	$6.98	2	http://www.pololu.com/product/1090/	$13.96	
Chassis – Motor Mounting Brackets (supplied in pairs)	Pololu	1164	$4.99	2	http://www.pololu.com/product/1089/	$9.98	
Battery – Lithium Polymer 1300 mAh 7.4V w/JST conne	MaxAmps	N/A	$31.49	1	http://www.maxamps.com/Lipo-1300-74-Pack.htm	$31.49	
Battery – Battery Connector Cord – JST Female	MaxAmps	N/A	$1.29	1	http://www.maxamps.com/JST-Female.htm	$1.29	
Battery – Charger Connector – JST Female	MaxAmps	N/A	$5.49	1	http://www.maxamps.com/proddetail.php?prod=HYP-Charger-Lead&cat=212	$5.49	
Arduino Microcontroller	Arduino	A000073	$25.95	1	http://www.mouser.com/ProductDetail/Arduino/A000073/?qs=sGAEpiMZZMuRQRjHmt4ZIaX9Ks%2fmeRGc	$25.95	
Arduino Proto Shield	Arduino	A000082	$4.95	1	http://www.mouser.com/ProductDetail/Arduino/A000082/?qs=BC3YYPaifMrP%252bNwhUyolJw==	$4.95	
4-pin Screw Terminal	Phoenix Contact	1725672	$2.43	3	http://www.mouser.com/ProductDetail/Phoenix-Contact/1725672/?qs=%2fha2pyFadug5do2T4X2cVsn4pI99S0GOyC	$7.29	
6-Pin Screw Terminal	Phoenix Contact	1725698	$3.64	1	http://www.mouser.com/ProductDetail/Phoenix-Contact/1725698/?qs=%2fha2pyFadug5do2T4X2cVg%252bYxJtujvF	$3.64	
8-Pin Screw Terminal	Phoenix Contact	1725711	$4.84	1	http://www.mouser.com/ProductDetail/Phoenix-Contact/1725711/?qs=%2fha2pyFadug5do2T4X2cVkQMjmkHXvU%2	$4.84	
1 3/8" Standoff – Aluminum M/F 4-40 1/4"	RAF Electronic Hardwar	4548-440-AL-7	$1.14	8	http://www.mouser.com/Search/ProductDetail.aspx?R=4548-440-AL-7virtualkey68520000virtualkey761-4548-440-AL-	$9.12	
7/16" Standoff – Aluminum M/F 4-40 3/16"	RAF Electronic Hardwar	4504-440-AL-7	$0.41	4	http://www.mouser.com/Search/ProductDetail.aspx?R=4504-440-AL-7virtualkey68520000virtualkey761-4504-440-AL-	$1.64	
Compass Module – HMC6352	Sparkfun	SEN-07915	$34.95	1	https://www.sparkfun.com/products/7915	$34.95	
Ardumoto Motor Driver Shield	Sparkfun	DEV-09815	$24.95	1	https://www.sparkfun.com/products/9815	$24.95	
Socket Head Cap Screw – 4-40 1/4" Stainless	McMaster-Carr	92185A106	$5.37	3	Box of 50 – go to www.mcmaster.com and enter part number, direct links don't work	$0.32	
Socket Head Cap Screw – 2-56 1/2" Stainless	McMaster-Carr	92185A083	$3.63	8	Box of 25	$1.16	
Flat Washer – #2 Stainless	McMaster-Carr	92141A003	$1.40	8	Box of 100	$0.11	
Button Head Cap Screw – 4-40 3/4" Stainless	McMaster-Carr	91864A434	$5.12	5	Box of 50	$0.51	
Nylock Safety Nut – 4-40 Stainless	McMaster-Carr	918331A005	$3.73	3	Box of 100	$0.11	
Nylon Spacer for iPod Lid			$6.93	5	http://barnhillbolt.com/itDetail.asp?it=302-40-7-FW4-093	$0.35	0.93" thick - needed for 4th gen iPod not needed for 5th gen
Lynxmotion Aluminum Multi-Purpose Sensor Housing	Lynxmotion	MPSH-01	$4.95	1	http://www.robotshop.com/en/lynxmotion-multi-purpose-sensor-housing.html	$4.95	
Devantech SRF05 Ultrasonic Rangefinder	Devantech	SRF05	$27.95	1	http://www.robotshop.com/en/devantech-ultrasonic-range-finder-srf05.html	$27.95	
Redpark iPhone to TTL Serial Adapter	Redpark	RB-Red-02	$54.58	1	http://www.robotshop.com/en/redpark-iphone-ttl-serial-adapter.html	$54.58	If using 5th gen iPod with Lightning connector then use dedicated Redpark Cable and RS-232 to Serial Adapter – Pololu Part #127 (partially
Laser Cut Chasiss	Pololu	N/A		1	Custom Laser Cutting Service – use current .DXF files for following parts – Cost is approximate for all parts	$45.00	0.250 Black Acrylic
Laser Cut iPod Base				1			0.125 Black Acrylic
Laser Cut iPod Cradle				1			0.250 Black Acrylic
Laser Cut iPod Lid				1			0.065 Clear Acrylic
Laser Cut Forward Facing Mirror				1			0.125 Mirror Acrylic
Laser Cut Motor Cover				4			0.065 Clear Acrylic
iPod Touch – 32GB Model (4th Generation - Refurbishe	Apple	N/A	$179.00	1	http://store.apple.com/us/product/FC544LL/A/refurbished-ipod-touch-32gb-black	$179.00	
					TOTAL	$493.58	Cost is approximate since laser cutting price may vary depending on model
Additional							
Lithium Polymer Battery Charger	Thunder		$119.00	2	http://www.hobbypartz.com/73p-t6x4-charger.html	$238.00	One charger per four iAnts
Battery Charger Power Supply	Thunder		$59.00	2	http://www.hobbypartz.com/73p-t350-power.html	$118.00	One PS Per Charger
Powered USB Hub							
Light Source For Nest							
Wireless Router for Server Connection							
Apple Computer for Server / GA / Programming							

Figure C.1: iAnt Complete Parts List

References

[1] J. P. Hecker and M. E. Moses, "Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms," *Swarm Intelligence*, vol. 9, no. 1, pp. 43–70, 2015.

[2] J. P. Hecker, K. Letendre, K. Stolleis, D. Washington, and M. E. Moses, "Formica ex machina: Ant swarm foraging from physical to virtual and back again," in *Swarm Intelligence: 8th International Conference, ANTS 2012*. Berlin, DE: Springer Berlin Heidelberg, 2012, pp. 252–259.

[3] D. Goldberg and M. J. Mataric, "Design and evaluation of robust behavior-based controllers for distributed multi-robot collection tasks," in *Robot teams: From diversity to polymorphism*. Citeseer, 2001.

[4] W. Liu, A. F. Winfield, and J. Sa, "Modelling swarm robotic systems: A case study in collective foraging," *Towards Autonomous Robotic Systems (TAROS 07)*, pp. 25–32, 2007.

[5] F. Duvallet, J. Kong, E. Marinelli, K. Woo, A. Buchan, B. Coltin, C. Mar, and B. Neuman, "Developing a low-cost robot colony," in *AAAI Fall Symposium*, 2007.

[6] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[7] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3293–3298.

[8] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proceedings of the 9th conference on autonomous robot systems and competitions*, vol. 1, no. LIS-CONF-2009-004. IPCB: Instituto Politécnico de Castelo Branco, 2009, pp. 59–65.

[9] A. F. Winfield, "Foraging robots," in *Encyclopedia of Complexity and Systems Science*. Springer, 2009, pp. 3682–3700.

[10] M. J. Mataric, "Learning to behave socially," in *Third international conference on simulation of adaptive behavior*, vol. 617. Citeseer, 1994, pp. 453–462.

[11] J. Pugh and A. Martinoli, "Inspiring and modeling multi-robot search with particle swarm optimization," in *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*. IEEE, 2007, pp. 332–339.

[12] A. Yershova, L. Jaillet, T. Siméon, and S. M. LaValle, "Dynamic-domain rrts: Efficient exploration by controlling the sampling domain," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 3856–3861.

[13] B. Burns and O. Brock, "Single-query motion planning with utility-guided random trees," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 3307–3312.

[14] L. Jaillet, A. Yershova, S. M. La Valle, and T. Siméon, "Adaptive tuning of the sampling domain for dynamic-domain rrts," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 2851–2856.

[15] J. W. Moore and M. Swerdling, "Conceptual design of a 1979 mars rover," 1971.

[16] A. M. Thompson, "The navigation system of the jpl robot," 1977.

[17] R. Washington, K. Golden, J. Bresina, D. E. Smith, C. Anderson, and T. Smith, "Autonomous rovers for mars exploration," in *Aerospace Conference, 1999. Proceedings. 1999 IEEE*, vol. 1. IEEE, 1999, pp. 237–251.

[18] J. H. Reif, "Complexity of the movers problem and generalizations extended abstract," in *Proceedings of the 20th Annual IEEE Conference on Foundations of Computer Science*, 1979, pp. 421–427.

[19] J. L. Crowley, "Navigation for an intelligent mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 1, no. 1, pp. 31–41, 1985.

[20] J. Ng and T. Bräunl, "Performance comparison of bug navigation algorithms," *Journal of Intelligent and Robotic Systems*, vol. 50, no. 1, pp. 73–84, 2007.

[21] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1-4, pp. 403–430, 1987.

[22] C. H. Chiang, J.-S. Liu, and Y.-S. Chou, "Comparing path length by boundary following fast matching method and bug algorithms for path planning," in *Opportunities and Challenges for Next-Generation Applied Intelligence*. Springer, 2009, pp. 303–309.

[23] J. Antich, A. Ortiz, and J. Minguez, "A bug-inspired algorithm for efficient anytime path planning," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 5407–5413.

[24] N. Buniyamin, W. Wan Ngah, N. Sariff, and Z. Mohamad, "A simple local path planning algorithm for autonomous mobile robots," *International journal of systems applications, Engineering & development*, vol. 5, no. 2, pp. 151–159, 2011.

[25] M. Zohaib, S. M. Pasha, N. Javaid, and J. Iqbal, "Intelligent bug algorithm (iba): A novel strategy to navigate mobile robots autonomously," *arXiv preprint arXiv:1312.4552*, 2013.

[26] Y. Zhu, T. Zhang, J. Song, and X. Li, "A new bug-type navigation algorithm considering practical implementation issues for mobile robots," in *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*. IEEE, 2010, pp. 531–536.

[27] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee, "Pheromone robotics," *Autonomous Robots*, vol. 11, no. 3, pp. 319–324, 2001.

[28] R. Mayet, J. Roberz, T. Schmickl, and K. Crailsheim, "Antbots: A feasible visual emulation of pheromone trails for swarm robots," in *Swarm Intelligence*. Springer, 2010, pp. 84–94.

[29] M. Krieger, J. Billeter, and L. Keller, "Ant-like task allocation and recruitment in cooperative robots," *Nature*, vol. 406, pp. 992–995, 2000.

[30] R. Russell, "Ant trails-an example for robots to follow?" in *Proceedings of the 1999 IEEE International Conference onRobotics and Automation*, vol. 4. IEEE, 1999, pp. 2698–2703.

[31] S. Garnier, F. Tache, M. Combe, A. Grimal, and G. Theraulaz, "Alice in pheromone land: An experimental setup for the study of ant-like robots," in *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*. IEEE, 2007, pp. 37–44.

[32] T. Schmickl and K. Crailsheim, "Trophallaxis within a robotic swarm: bio-inspired communication among robots in a swarm," *Autonomous Robots*, vol. 25, no. 1, pp. 171–188, 2008.

[33] A. Dussutour, S. C. Nicolis, G. Shephard, M. Beekman, and D. J. Sumpter, "The role of multiple pheromones in food recruitment by ants," *The Journal of Experimental Biology*, vol. 212, no. 15, pp. 2337–2348, 2009.

[34] D. J. Sumpter and M. Beekman, "From nonlinearity to optimality: pheromone trail foraging by ants," *Animal behaviour*, vol. 66, no. 2, pp. 273–280, 2003.

[35] D. E. Jackson and F. L. Ratnieks, "Communication in ants," *Current biology*, vol. 16, no. 15, pp. R570–R574, 2006.

[36] K. Holder and G. Polis, "Optimal and central-place foraging theory applied to a desert harvester ant, pogonomyrmex californicus," *Oecologia*, vol. 72, no. 3, pp. 440–448, 1987.

[37] B. Beverly, H. McLendon *et al.*, "How site fidelity leads to individual differences in the foraging activity of harvester ants," *Behavioral Ecology*, vol. 20, no. 3, pp. 633–638, 2009.

[38] T. Paz Flanagan, K. Letendre, W. Burnside, G. M. Fricke, and M. Moses, "How ants turn information into food," in *Artificial Life (ALIFE), 2011 IEEE Symposium on*. IEEE, 2011, pp. 178–185.

[39] F. Steele Jr and G. Thomas, "Directed stigmergy-based control for multi-robot systems," in *Proceedings of the ACM/IEEE international conference on Human-robot interaction*. ACM, 2007, pp. 223–230.

[40] S. M. LaValle and J. J. Kuffner Jr, "Rapidly-exploring random trees: Progress and prospects," 2000.

[41] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[42] N. R. Hoff, A. Sagoff, R. J. Wood, and R. Nagpal, "Two foraging algorithms for robot swarms using only local communication," in *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*. IEEE, 2010, pp. 123–130.

[43] M. Dorigo, D. Floreano *et al.*, "Swarmanoid: a novel concept for the study of heterogeneous robotic swarms," Technical Report TR/IRIDIA/2011-014, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, Tech. Rep., 2011.

[44] P. Bailis, R. Nagpal, and J. Werfel, "Positional communication and private information in honeybee foraging models," *Swarm Intelligence*, pp. 263–274, 2010.

[45] A. Campo and M. Dorigo, "Efficient multi-foraging in swarm robotics," in *Advances in Artificial Life*. Springer, 2007, pp. 696–705.

[46] I. Kelly, O. Holland, and C. Melhuish, "Slugbot: A robotic predator in the natural world," in *Proceedings of the Fifth International Symposium on Artificial Life and Robotics for Human Welfare and Artificial Liferobotics*. Citeseer, 2000, pp. 470–475.

[47] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[48] Y. Mohan and S. Ponnambalam, "An extensive review of research in swarm robotics," in *World Congress on Nature & Biologically Inspired Computing*. IEEE, 2009, pp. 140–145.

[49] T. P. Flanagan, K. Letendre, W. R. Burnside, G. M. Fricke, and M. E. Moses, "Quantifying the effect of colony size and food distribution on harvester ant foraging," *PloS one*, vol. 7, no. 7, p. e39427, 2012.

[50] K. Letendre and M. E. Moses, "Synergy in ant foraging strategies: memory and communication alone and in combination," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013, pp. 41–48.

[51] J. P. Hecker and M. E. Moses, "An evolutionary approach for robust adaptation of robot behavior to sensor error," in *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '13 Companion)*. New York, NY: ACM, 2013, pp. 1437–1444. [Online]. Available: http://doi.acm.org/10.1145/2464576.2482724

[52] J. P. Hecker, K. Stolleis, B. Swenson, K. Letendre, and M. E. Moses, "Evolving error tolerance in biologically-inspired iAnt robots," in *Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems (Advances in Artificial Life, ECAL 2013)*. Cambridge, MA: MIT Press, 2013, pp. 1025–1032.

[53] T. O. Crist and J. W. Haefner, "Spatial model of movement and foraging in harvester ants (pogonomyrmex)(ii): the roles of environment and seed dispersion," *Journal of Theoretical Biology*, vol. 166, no. 3, pp. 315–323, 1994.

[54] A. Johnson, J. Wiens, B. Milne, and T. Crist, "Animal movements and population dynamics in heterogeneous landscapes," *Landscape ecology*, vol. 7, no. 1, pp. 63–75, 1992.

[55] M. Zohaib, M. Pasha, R. Riaz, N. Javaid, M. Ilahi, and R. Khan, "Control strategies for mobile robot with obstacle avoidance," *arXiv preprint arXiv:1306.1144*, 2013.

[56] A. J. Denny, J. Wright, and B. Grief, "Foraging efficiency in the wood ant, formica rufa: is time of the essence in trail following?" *Animal Behaviour*, vol. 62, no. 1, pp. 139–146, 2001.